

This item was submitted to Loughborough's Institutional Repository (<https://dspace.lboro.ac.uk/>) by the author and is made available under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

Loughborough University Institutional Repository

A manufacturing core concepts ontology to support knowledge sharing

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: <https://dspace.lboro.ac.uk/2134/9857>

Publisher: © Zahid Usman

Please cite the published version.

A Manufacturing Core Concepts Ontology to Support Knowledge Sharing

By

Zahid Usman

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of
Doctor of Philosophy of Loughborough University

May 2012



© by Zahid Usman (2012)



CERTIFICATE OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgments or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a degree.

..... (Signed)

Zahid Usman

..... (Date)

Abstract

Knowledge sharing across domains is key to bringing down the cost of production and the time to market of products. This thesis is directed to improve the knowledge sharing capability of the present systems that use information and communication technologies. Systems for different domains have structures that are made up of concepts and relations with different semantic interpretations. Therefore, knowledge sharing across such domains becomes an issue. Knowledge sharing across multiple domains can be facilitated through a system that can provide a shared understanding across multiple domains. This requires a rigorous common semantic base underlying the domains across which to share knowledge.

In this thesis, a Manufacturing Core Concepts Ontology is proposed as a common semantic base to support knowledge sharing across manufacturing domains. The particular focus is to capture production knowledge and share it with product design. To achieve this, a set of core manufacturing concepts and relations have been defined. Formal i.e. computer understandable logic is used to capture the semantics of concepts. An approach to gradually specialize concepts at different levels has been proposed to capture the variations in the depths of meaning of concepts. Within the proposed ontology, a set of concepts and a methodology has been defined to enable capturing and reasoning about the production knowledge at multiple levels of abstraction. The proposed ontology and approach supports the development of semantically sound application specific product design domain and production domain ontologies. These ontologies can be linked for knowledge sharing through the semantic base provided by the manufacturing core concepts ontology.

A detailed experimental investigation has been conducted to verify the ontology. It has been shown experimentally that the semantics of the concepts and the varying depths of meaning of those concepts have been formally captured such that the computer systems can understand the semantics and respond accordingly. The proposed ontology has been shown to support the development of semantically sound application specific product design and production ontologies and provide a route to knowledge sharing across them. It has been made possible to capture and reason about the production methods at multiple levels of knowledge abstractions, which goes beyond the capability of objected oriented systems.

Keywords: ontologies, product lifecycle management, semantics, manufacturing core concepts, interoperability, knowledge sharing, features and part family.

Acknowledgements

Many people and organisations have, directly or indirectly, contributed towards this thesis. I would like to thank all of them for their support, contribution and help in achieving my PhD.

This PhD was funded by the Innovative Manufacturing and Construction Research Centre (IMCRC) under the Interoperable Manufacturing Knowledge Systems (IMKS) project and the Wolfson School of Mechanical and Manufacturing Engineering of Loughborough University.

First of all I would like to thank almighty **Allah** who made it possible for me to achieve this milestone. Then I wish to extend my utmost gratitude to my supervisor, Dr. Bob Young who has been like a father figure throughout my PhD. Dr. Young is an expert in his field, possesses up to date knowledge and is a great guide. He always gave me plenty of time even during the busiest of his schedules. Dr. Young's critical thinking and analysis helped me focus on the finer details of research work. During the research project he maintained a fine balance between the project and my PhD. Dr. Young's supervision not only helped me develop as a researcher but it also enhanced my learning of life in general. Without his kind supervision, this PhD would not have been possible.

I would also like to thank my co supervisor Prof. Keith Case for his useful input in several meetings and for providing detailed feedback during my thesis write-up.

I would like to take this opportunity to specially thank Dr. Nitishal Chungoora (Tish) who has been an ever available and reliable source of encouragement, knowledge and guidance throughout my PhD. Tish has contributed a lot to my PhD particularly in assisting with the technical details, in writing up of the thesis and publishing research papers. I would also like to thank Dr. George Gunendran who always helped me with technical as well as administrative issues.

I would like to thank Dr. Claire Palmer, Dr. J.A. Harding, Dr. N.A. Anjum and Mr. Muhammad Imran for their assistance, support and guidance. I also wish to thank Prof. Anne-Françoise Cutting-Decelle and Prof. J. Gao for their guidance in project meetings.

I also wish to thank everyone from Rolls Royce Plc. who supported me during the industrial secondment. Special thanks to Mr. Mark Heyman, Mr. Kamran Shahzad to whom I was directly reporting. My kind appreciation also goes to my friends and colleagues Farukh, Atta, Tanveer Nazeer (Polla), Dr. Tanveer, Ammar (Malak), Dr. Mubashar, Arina and my housemates Yasir, Farhan, Amjad and Younus for their support.

My kind regards also goes to my University of Engineering and Technology (UET) Lahore and the colleagues at UET, Dr. Ijaz A. Choudhry, Mr. Ziaullah Khan and everyone else who has been supportive. I also wish to thank the ex chairperson of the Higher Education Commission (HEC) of Pakistan, Prof. Dr. Ata-ur-Rehman, for it was his vision, effort and guidance that lead me and many of fellow Pakistanis towards the higher studies abroad.

I also wish to thank everyone who contributed directly or indirectly to make this PhD possible and whose names I have not been able to mention.

Dedication

I dedicate this work, before anyone else, to my beloved Holy Prophet Hazrat **Muhammad** bin Abdullah (may peace and blessing of Allah be upon him and his family and friends) as an ever insufficient token of my gratitude.

After that, I wish to dedicate this thesis to my mother her Highness Safeen Akhtar, who has always been a constant source of prayers, unconditional love and a never ending support. No words can justify her stature and her contribution. I also dedicate this work to my dad Mr. Muhammad Usman who has always been proud of us and has a heart of gold.

I also dedicate this work to my grandmother Ghulam Fatima who passed away during my undergraduate studies, and I always miss that continuous source of prayers and motivation. I also wish to dedicate this thesis to my grandfather Ghulam Muhammad, whom I never met but I have the utmost respect and honour for him. I would also like to dedicate this work to Chacha Abdul Ghaffar Khan who has worked tirelessly and selflessly till this day to see me and the whole family happy. I also dedicate this to my brother Shahid Usman in particular, and the rest of my siblings in general. Also to my siblings Fahad, Ali and Farah who are no more.

Finally I wish to dedicate this thesis to Hazrat Wasif Ali Wasif (May Allah shower his blessing upon him) who has become a guide and inspiration for me.

CONTENTS

ABBREVIATIONS.....	XIII
1 INTRODUCTION.....	1
1.1 RESEARCH CONTEXT	1
1.2 AN OVERVIEW OF THE PROPOSED APPROACH	2
1.3 AIMS AND OBJECTIVES	3
1.4 RESEARCH METHODOLOGY.....	4
1.4.1 Hypothesis	4
1.4.2 Ontology Development Methodology	4
1.5 RESEARCH SCOPE.....	6
1.5.1 IMKS Project	6
1.5.2 Ontology Development Languages and Tools.....	7
1.5.2.1 Ontology Development Languages.....	7
1.5.2.2 Ontology Development Tools.....	7
1.5.3 Focused Production Area	8
1.6 THESIS STRUCTURE.....	8
2 ONTOLOGY BASED KNOWLEDGE SHARING SYSTEMS	10
2.1 INTRODUCTION	10
2.2 A GENERAL VIEW OF KNOWLEDGE SHARING AND INTEROPERABILITY.....	10
2.2.1 Definitions of Data, Information and Knowledge.....	10
2.2.2 Levels of Knowledge	11
2.2.2.1 Shallow Knowledge	11
2.2.2.2 Deep Knowledge	12
2.2.3 Types of Knowledge.....	12
2.2.3.1 Types of Knowledge with respect to Explicitness	12
2.2.3.2 Types of Knowledge with respect to Configuration	12
2.2.4 Manufacturing Knowledge	13
2.2.5 Knowledge sharing as an Issue in Manufacturing	13
2.2.6 Interoperability.....	14
2.3 METHODS TO ACHIEVE INTEROPERABILITY.....	15
2.3.1 Model Driven Interoperability (MDI).....	15
2.3.2 Frameworks for interoperability.....	18
2.3.2.1 IDEAS Interoperability Framework	18
2.3.2.2 ATHENA Interoperability Framework (AIF).....	18
2.3.2.3 Standard on Frameworks for Interoperability.....	19
2.3.2.3.1 Integrated Approach.....	20
2.3.2.3.2 Unified Approach	20
2.3.2.3.3 Federated Approach.....	21
2.4 ONTOLOGICAL METHODS FOR INTEROPERABLE SYSTEMS	22
2.4.1 Definition of ‘Ontology’.....	22
2.4.2 Types of Ontologies	23
2.4.2.1 Types of Ontologies with respect to Formalisation	24
2.4.2.1.1 Lightweight Ontologies.....	24
2.4.2.1.2 Heavyweight Ontologies	24
2.4.2.2 Types of Ontologies with respect to the Specificity	24
2.4.2.2.1 Foundation ontologies	24
2.4.2.2.2 Core Ontologies	25
2.4.2.2.3 Domain Ontologies	26
2.4.3 Ontology Development Methodologies.....	26
2.4.3.1 IDEF5 Ontology Development Methodology	26
2.4.3.2 Blomqvist and Ohgren’s Methodology.....	27
2.4.3.3 Noy and McGuinness Methodology	29
2.4.3.4 METHONTOLOGY	29
2.4.3.5 CommonKADS Methodology	30
2.4.3.6 Review of Other Ontology Development Methodologies.....	30
2.4.3.7 Mapping and Merging of Ontologies	31

2.4.4	Ontology Development Languages and Tools.....	32
2.4.4.1	Ontology Development Languages.....	32
2.4.4.1.1	Unified Modelling Language (UML).....	33
2.4.4.1.2	HTML & XML.....	34
2.4.4.1.3	RDF & RDF(S).....	34
2.4.4.1.4	Web Ontology Language (OWL).....	35
2.4.4.1.5	Common Logic.....	35
2.4.4.2	Ontology Development Tools.....	36
2.4.4.2.1	IODE & XKS.....	36
2.4.5	Purpose and Uses of Ontology.....	37
2.5	ONTOLOGIES IN MANUFACTURING.....	37
2.5.1	Manufacturing Enterprise Level Ontologies.....	39
2.5.1.1	Lightweight Manufacturing Enterprise Level Ontologies.....	39
2.5.1.1.1	CIM Open System Architecture (CIMOSA).....	39
2.5.1.1.2	Factory Design Model (FDM).....	39
2.5.1.1.3	The Use of Standards as Ontologies.....	40
2.5.1.1.4	Ontological Integration of Product Lifecycle Knowledge.....	40
2.5.1.2	Heavyweight Manufacturing Enterprise Level Ontologies.....	41
2.5.1.2.1	Enterprise Project and TOVE Project.....	41
2.5.1.2.2	Manufacturing Systems Engineering (MSE) Ontology.....	41
2.5.1.2.3	Process Specification Language.....	42
2.5.1.2.4	Semantic Interoperability between Application Ontologies.....	43
2.5.2	Detailed Manufacturing Ontologies.....	43
2.5.2.1	Lightweight Detailed Manufacturing Ontologies.....	43
2.5.2.1.1	Model Oriented Simultaneous Engineering System (MOSES).....	43
2.5.2.1.2	Holonic Approach to Manage Product Lifecycle Data.....	44
2.5.2.1.3	A Product Ontology for Integrating Production Planning and Design.....	44
2.5.2.1.4	A Model to Share Manufacturing Best Practice Knowledge.....	45
2.5.2.1.5	The Core Product Model.....	45
2.5.2.1.6	An Ontology Based Tool for Product Data Exchange.....	46
2.5.2.2	Heavyweight Detailed Manufacturing Ontologies.....	46
2.5.2.2.1	Manufacturing Semantics Ontology (MASON).....	46
2.5.2.2.2	A Machining Ontology Based on MASON.....	47
2.5.2.2.3	ADACOR ontology.....	47
2.5.2.2.4	Ontology Based Multilayer Knowledge Framework.....	48
2.5.2.2.5	Design for Manufacturing (DFM) ontology.....	49
2.5.2.2.6	An Ontology for Integration of CAD and CAPP Systems.....	49
2.5.2.2.7	Semantic Manufacturing Interoperability Framework.....	50
2.5.3	Feature Based ontologies.....	51
2.5.3.1	Overview of Feature Based Technology.....	51
2.5.3.2	Feature Based View on Interoperability between Design and Production.....	52
2.5.3.3	The Use of Predefined Standard Features for Interoperability.....	52
2.5.3.4	Mapping Different Features for Interoperability.....	53
2.5.4	Non Manufacturing Ontologies of Direct Relevance.....	55
2.6	SUMMARY.....	55
3	AN INDUSTRIAL STUDY OF MANUFACTURING CONCEPTS.....	58
3.1	INTRODUCTION.....	58
3.2	INVESTIGATION OF DISC DESIGN AND PRODUCTION SYSTEMS.....	58
3.2.1	Designer's Perspective of Disc.....	59
3.2.1.1	Exploration of Disc Design Features.....	59
3.2.1.2	Exploration of Disc Design Part Families.....	60
3.2.2	Production Engineer's Perspective of the Disc.....	61
3.2.2.1	Knowledge Abstraction in Disc Production Methods.....	62
3.2.2.2	Exploration of Disc Production Features.....	63
3.2.2.3	Exploration of Disc Production Part Families.....	64
3.3	RELATIONS BETWEEN DESIGN AND PRODUCTION.....	65

3.3.1	Need to Relate Product Design and Production Concepts.....	65
3.3.2	Relating Design and Production for Knowledge Sharing	67
3.3.2.1	Example: Relating Design & Production Features for Knowledge Sharing	67
3.4	SUMMARY	69
4	THE REQUIREMENTS OF A MANUFACTURING CORE CONCEPTS ONTOLOGY	71
4.1	INTRODUCTION	71
4.2	RESEARCH ISSUES IN MANUFACTURING ONTOLOGIES.....	71
4.3	IDENTIFYING CONCEPTS & RELATIONS AND FORMALISING THEIR SEMANTICS.....	73
4.4	ISSUE OF CAPTURING VARYING DEPTHS OF MEANING OF CONCEPTS	76
4.5	CAPTURING AND REASONING AT MULTIPLE LEVELS OF KNOWLEDGE ABSTRACTION	78
4.5.1	Knowledge Abstraction Levels: The Meta and Instance Relations	78
4.5.1.1	Individual Level Knowledge	79
4.5.1.2	Meta Level Knowledge	79
4.5.2	Knowledge Abstraction Levels in Production:.....	80
4.6	SUMMARY	82
5	MANUFACTURING CORE CONCEPTS & RELATIONS AND THEIR FORMALISATION	83
5.1	INTRODUCTION	83
5.2	EXPLORATION OF CORE CONCEPTS AND INTRA-CATEGORY RELATIONS.....	84
5.2.1	Realised Part.....	85
5.2.1.1	Realised Part concepts.....	85
5.2.1.2	Realised Part's Intra-Category Relations.....	87
5.2.2	Part Version	88
5.2.2.1	Part Version Concepts.....	88
5.2.2.2	Part Version's Intra-Category Relations	89
5.2.3	Manufacturing Resource.....	89
5.2.3.1	Manufacturing Resource concepts.....	89
5.2.3.2	Manufacturing Resource's Intra-category Relations	92
5.2.4	Manufacturing Facility.....	92
5.2.4.1	Manufacturing Facility Concepts	92
5.2.4.2	Manufacturing Facility's Intra-Category Relations.....	93
5.2.5	Manufacturing Method.....	94
5.2.5.1	Production Method Concepts.....	94
5.2.5.2	Production Method's Intra-Category Relations	96
5.2.6	Manufacturing Process.....	96
5.2.7	Feature	97
5.2.8	Part Family.....	97
5.2.8.1	Part Family Concepts	97
5.2.8.2	Part Family's Intra-Category Relations.....	98
5.3	INTER-CATEGORY RELATIONS	99
5.3.1	Realised Part's Inter-category Relations	99
5.3.1.1	With Part Version Concepts.....	99
5.3.1.2	With Manufacturing Method Concepts	99
5.3.1.3	With Manufacturing Facility Concepts	99
5.3.2	Part Version's Inter-category Relations with Other Categories	100
5.3.2.1	With Manufacturing Method Concepts	100
5.3.3	Manufacturing Facility's Inter-category Relations	100
5.3.3.1	With Manufacturing Process Concepts	100
5.3.3.2	With Manufacturing Method Concepts	100
5.3.4	Manufacturing Resource's Inter-Category Relations	100
5.3.4.1	With Manufacturing Method Concepts	100
5.3.5	Feature's Inter-category Relations	101
5.3.6	Part Family's Inter-Category Relations.....	101
5.3.6.1	With Manufacturing Method Concepts	102
5.3.6.2	With Feature Concepts.....	102
5.3.7	Manufacturing Method's Inter-Category Relations.....	102
5.3.7.1	With Manufacturing Process Concepts	102
5.4	COMBINED LIGHTWEIGHT REPRESENTATION OF THE MCCO.....	102
5.5	HEAVYWEIGHT FORMALISATION OF THE MCCO	104

5.5.1	The Use of KFL for Heavyweight Formalisation	104
5.5.1.1	Declaring Concepts, Relations and Function.....	104
5.5.1.2	Axiomatization.....	105
5.5.2	An example: Heavyweight Formalisation of PartFamily Concepts	106
5.5.2.1	Formalisation of the Concept PartFamily.....	106
5.5.2.2	Formalisation of the Concept ProductionPartFamily	107
5.6	SUMMARY	107
6	SPECIALIZING CONCEPTS TO CAPTURE THE VARYING DEPTHS OF MEANING	109
6.1	INTRODUCTION	109
6.2	VARIATION OF DEPTHS OF MEANING WITHIN THE MCCO.....	109
6.3	AN OVERVIEW OF SPECIALIZATION LEVELS	110
6.3.1	Generic Core Concepts Level.....	111
6.3.2	Product Lifecycle Generic Core Concepts Level	112
6.3.3	Product Design and Production Core Concepts Levels	112
6.3.3.1	Product Design Core Concepts Level	112
6.3.3.2	Production Core Concepts Level	113
6.4	AN EXAMPLE: SPECIALIZATION LEVELS EXPLAINED USING FEATURE CONCEPTS	113
6.4.1	Generic Level Feature Concepts	114
6.4.1.1	'Feature' Concept.....	114
6.4.1.2	Formalization of 'Feature' Concept	114
6.4.1.3	FormFeature Concept	115
6.4.1.4	Formalization of FormFeature.....	116
6.4.2	Product Lifecycle Generic Feature Concept.....	117
6.4.2.1	Formalization of the Concept 'PartFeature'	117
6.4.3	Design and Production level Feature Concepts.....	118
6.4.3.1	Design Level Feature Core Concept.....	118
6.4.3.2	Formalization of 'DesignFeature'	119
6.4.3.3	Production Level Feature Core Concepts.....	120
6.4.3.4	Formalization of 'ProductionFeature'.....	120
6.5	DEVELOPMENT OF SPECIALIZED ONTOLOGIES USING FEATURE CONCEPTS	121
6.6	ROUTE TO KNOWLEDGE SHARING THROUGH THE FEATURE CONCEPTS	122
6.6.1.1	An Example of Relating Design and Production Feature(s)	123
6.7	SUMMARY	124
7	THE USE OF 'CLABJECT' TO CAPTURE AND REASON ABOUT META LEVEL PRODUCTION METHODS.....	126
7.1	INTRODUCTION	126
7.2	REQUIREMENTS FOR MULTIPLE LEVELS OF KNOWLEDGE ABSTRACTIONS.....	126
7.3	KNOWLEDGE CAPTURE AND REASONING AT META LEVEL	128
7.3.1	Capturing Meta level Feature and PartFamily Production Methods	128
7.3.1.1	The Need to Define the <i>MetaMeta level</i> Structure for the <i>Meta level Knowledge</i>	128
7.3.1.2	The Use of 'Clabjects' and 'Powertypes'	129
7.3.2	Capturing Sequencing of the Relation 'minPrecedes'	132
7.3.2.1	Lightweight Formalisation of 'minPrecedes'	133
7.3.2.2	Heavyweight Formalisation of 'minPrecedes'	134
7.3.3	Reasoning about Feature and PartFamily Production Methods.....	136
7.3.3.1	Reasoning About the Manufacturability of Stages in Operations.....	136
7.3.3.2	Logical Reasoning to Ensure Correct Sequencing	139
7.3.3.2.1	Formalising the Semantics of 'canBeAccommodatedIn'	141
7.3.4	Complete Lightweight Representation of ProductionMethod	143
7.4	SUMMARY	144
8	THE EXPERIMENTAL VERIFICATION OF THE RESEARCH CONCEPT	145
8.1	INTRODUCTION	145
8.2	METHOD OF IMPLEMENTATION AND EXPERIMENTAL VERIFICATION.....	145
8.3	IMPLEMENTATION OF THE MCCO IN IODE.....	146
8.4	OVERVIEW OF THE EXPERIMENTS	148
8.5	EXPERIMENT1: TESTING THE SPECIALIZATION LEVELS AND THE CAPTURE OF SEMANTICS.....	149

8.5.1	Objective.....	149
8.5.2	Explanation	149
8.5.3	Procedure	150
8.5.4	Discussion on Results	152
8.5.5	Conclusions	153
8.6	EXPERIMENT 2: DEVELOPING APPLICATION SPECIFIC ONTOLOGIES USING THE MCCO.....	153
8.6.1	Objectives	153
8.6.2	Explanation	153
8.6.3	Procedure	153
8.6.4	Formalisation and Implementation of Application Specific Ontologies.....	154
8.6.4.1	Verification of the Semantics in the Application Specific Ontologies	155
8.6.5	Discussion on Results	156
8.6.6	Conclusion	156
8.7	EXPERIMENT 3: TO SHOW THE PRODUCTION CONSEQUENCES OF CHANGING THE DESIGN OF A FEATURE	157
8.7.1	Procedure	157
8.7.1.1	The Capture of Production Knowledge.....	157
8.7.1.2	Establishing the Route to Knowledge Sharing.....	159
8.7.1.3	Getting Feedback from Production into Design	160
8.7.2	Discussion on Results	161
8.8	EXPERIMENT 4: TESTING THE CAPTURE OF AND REASONING ABOUT THE META LEVEL PRODUCTIONMETHODS	162
8.8.1	Objectives	162
8.8.2	Overview and procedure.....	162
8.8.2.1	Acquisition of PartFamilyProductionMethod(s) at the Meta Level	163
8.8.2.2	Acquisition of the FeatureProductionMethod at the Meta Level	164
8.8.2.3	Reasoning about the Manufacturability of MetaFeatureProductionMethod in PartFamilyProductionMethod.....	165
8.8.3	Discussion and Conclusions	166
8.9	CASE STUDY	167
8.9.1.1	The Capture of Production Knowledge.....	167
8.9.1.2	Establishing the Route to Knowledge Sharing.....	168
8.9.1.3	Getting Feedback from Production into Design	169
8.9.2	Discussion and Conclusions	170
8.10	SUMMARY	170
9	DISCUSSION, CONCLUSIONS AND FUTURE RESEARCH	172
9.1	INTRODUCTION	172
9.2	DISCUSSION	172
9.2.1	The MCCO as an Intermediate Set of Concepts	172
9.2.2	Capturing the Varying Depths of Meaning of Concepts	174
9.2.3	Developing Application Specific Ontologies	175
9.2.4	The Route to Knowledge Sharing between Design and Production	176
9.2.5	Capturing and Reasoning about Meta and Higher Levels of Knowledge	176
9.2.6	Other Aspects of the Research	177
9.2.6.1	Extension of the MCCO	177
9.2.6.2	Broader Effectiveness of the MCCO	178
9.3	CONCLUSIONS	180
9.4	FUTURE RESEARCH	181
	PUBLICATIONS	183
	REFERENCES	185
	A1AN INTRODUCTION TO THE KNOWLEDGE FRAME LANGUAGE.....	206
A1.1	INTRODUCTION	206
A1.2	WHAT IS KFL?	206
A1.3	CREATING A KFL FILE	206
A1.3.1	Contexts.....	207
A1.3.2	Properties.....	207

A1.3.2.1 Useful Property Kinds	208
A1.3.3 Relations	208
A1.3.4 MetaProperty	209
A1.3.5 Intensional and Extensional Relations	209
A1.3.6 Functions	210
A1.4 DOCUMENTATION	211
A1.5 MORE ON DIRECTIVES	212
A1.6 AXIOMATISATION	212
A1.6.1 Variables	213
A1.6.2 Logical Operators	213
A1.6.2.1 Implication	213
A1.6.2.2 Conjunction	213
A1.6.2.3 Disjunction	214
A1.6.2.4 Negation	214
A1.6.2.5 Universal Quantification	214
A1.6.2.6 Existential Quantification	214
A2 FORMALISATION OF MCCO	216
A2.1 CONTEXTS FOR THE MCCO AND APPLICATION SPECIFIC ONTOLOGIES	216
A2.2 GENERIC LEVEL FORMALISATION	216
A2.2.1 Generic Core Concepts	216
A2.2.2 Generic Core Relations	217
A2.2.3 Generic Core Axioms	218
A2.3 PRODUCT LIFECYCLE GENERIC LEVEL FORMALISATION	218
A2.3.1 Product Lifecycle Core Concepts	218
A2.3.2 Product Lifecycle Core Relations	221
A2.3.3 Product Lifecycle Core Functions	222
A2.3.4 Product Lifecycle Core Axioms	223
A2.4 DESIGN AND PRODUCTION LEVEL FORMALISATION	226
A2.4.1 Production Level Formalisation	226
A2.4.1.1 Production Core Concepts	226
A2.4.1.2 Production Core Relations	227
A2.4.1.3 Production Core Axioms	228
A2.4.2 Design Level Formalisation	228
A2.4.2.1 Design Core Concepts	228
A2.4.2.2 Design Core Relations and Functions	228
A2.4.2.3 Design Core Axioms	229
A2.5 PRODUCTION METHOD FORMALISATION FOR KNOWLEDGE CAPTURE AND REASONING AT META AND INDIVIDUAL LEVELS OF KNOWLEDGE	229
A2.6 MCCO EXTENSIONS	240
A2.6.1 Manufacturing Resource Extension	240
A2.6.2 Manufacturing Process Extension	244
A2.7 SPECIFIC PRODUCT DESIGN AND PRODUCTION ONTOLOGIES	246
A2.7.1 Design specific Ontology 'Aero Engine Disc Design Ontology'	246
A2.7.2 Production specific Ontology 'Aero Engine Disc Production Ontology'	248

Table of Figures

Figure 1.1: An overview of the research framework	2
Figure 1.2: Ontology Development Methodology	5
Figure 1.3: The IMKS concept	6
Figure 1.4: Thesis structure	9
Figure 2.1: The influence of design on costs (True and Izzi 2002)	14
Figure 2.2: MDA approach to interoperability	16
Figure 2.3: Reference Model for MDI adapted from (Lemrabet et al. 2010)	17
Figure 2.4 IDEAS interoperability framework redrawn form Chen et al (2004)	18
Figure 2.5: ATHENA interoperability framework (AIF) redrawn from Berre et al (2007)	19
Figure 2.6: Frameworks for enterprise interoperability (ISO/CEN-11354 2008).....	19
Figure 2.7: Blomqvist and Ohgren's Ontology Development Methodology (2008)	27
Figure 2.8: Categories of ontology development languages	32
Figure 2.9: An object, attribute and value triplet of RDF	35
Figure 2.10: MSE ontology as a mediator between different enterprise ontologies	42
Figure 2.11: Main concepts of MASON reproduced from (Leimagnan et al. 2006)	46
Figure 2.12: Main concepts and architecture of ADACORE Manufacturing ontology reproduced form Brogo and Leitao (2007)	48
Figure 2.13: Process for data exchange between CAD and CAPP systems (Dartigues et al, 2007)	49
Figure 2.14: Semantic Interoperability Framework (Chungoora, 2010).....	50
Figure 2.15: Breakdown of feature concepts (Dartigues et al, 2007)	54
Figure 3.1: Selected engine disc for industrial investigation	58
Figure 3.2: An adaptation of design perspectives of the studied disc.....	60
Figure 3.3: Transition from Forged Disc (Raw Material) to finish machined disc	61
Figure 3.4: An adaptation of production perspectives of HPC disc	62
Figure 3.5: Example of extracting the production method of features and capturing their production knowledge	64
Figure 3.6: Mismatches between production design and production concepts	66
Figure 3.7: Finding Production Features encompassing the form of design feature 'Circumferential Groove'	68
Figure 3.8: Mechanism for sharing Production Knowledge into design through features	69
Figure 4.1: The intermediate set of concepts between foundational and domain specific concepts	74
Figure 4.2: A view of "Resource" concept and its varying interpretations	77
Figure 4.3: An example of detailed individual level process plan constituted of individuals or instances.....	80
Figure 4.4: An example of abstracted/ Meta level ProcessPlan consisting of abstract concepts	81
Figure 4.5: Depiction of multiple levels of abstractions through MachineTool concepts and individuals	81
Figure 5.1: Categories of manufacturing core concepts and relations	84
Figure 5.2: Subsumption of RealisedPart under concept Object	86
Figure 5.3: Examples of realised parts	86
Figure 5.4: Lightweight representation of RealisedParts concepts	87
Figure 5.5: Lightweight formalisation of PartVersion concepts and relations	88
Figure 5.6: Examples of ManufacturingResource concepts.....	89
Figure 5.7: Manufacturing Resources hierarchy adapted from (Leimagnan et al. 2006).....	90
Figure 5.8: Lightweight representation of ManufacturingResource concepts	90
Figure 5.9: Manufacturing Facility lightweight representation adapted from (Zhao et al, 1999) ...	93
Figure 5.10: Lightweight formalisation of ProcessPlan and FeatureProductionMethod	95
Figure 5.11: PartFamily concepts.....	98
Figure 5.12: Lightweight representation of Manufacturing Core Concepts Ontology (MCCO) ..	103
Figure 6.1: Schematic view of the specialization levels in the MCCO and how they help develop application specific ontologies.....	111
Figure 6.2: Varying specializations of feature concepts in everyday life	114
Figure 6.3: Lightweight representation of feature concept	115

Figure 6.4: Lightweight representation of FormFeature	116
Figure 6.5: Lightweight Product Lifecycle level specialization through Product feature concept ..	117
Figure 6.6: Lightweight formalization of Design level specialization of Feature as DesignFeature	119
Figure 6.7: Lightweight formalization of Production level specialization of Feature as ProductionFeature.....	120
Figure 6.8: Lightweight view of application specific ontologies developed from the MCCO and the instantiated disc design and production views of application specific ontologies.	122
Figure 6.9: Relating design and production features through their overlapping forms.....	123
Figure 6.10: Feature concepts across specializations level, lightweight representation	125
Figure 7.1: Requirements of capturing feature and Part Manufacturing Methods at Individual and Meta levels	127
Figure 7.2: Manufacturing methods for part at individual and Meta levels	129
Figure 7.3: Lightweight formalisation of ProcessPlan's powertype, clabjects and their individuals	130
7.4: Structure for the acquisition of meta level manufacturing method	132
Figure 7.5: Lightweight formalisation of relation 'minPrecedes'	133
Figure 7.6: a) Relation minPrecedes to capture sequencing of stages in a RimProductionMethod,	134
Figure 7.7: Illustration of the relation <i>canBeManufacturedIn</i>	137
7.8: Lightweight formalisation of relation 'canBeManufacturedIn'	138
Figure 7.9: True and False conditions for FeatureProductionMethod to be accommodated in a PartFamilyProductionMethod	140
Figure 7.10: Complete lightweight formalisation of ProductionMethod as part of MCCO	143
Figure 8.1: Scheme for the formalization of the MCCO and its experimental investigation	145
Figure 8.2: Implementation of the MCCO in IODE	147
Figure 8.3: The arrangement of experiments.....	148
Figure 8.4: Asserting a ProductionFeature without its ProductionMethod	150
Figure 8.5: Asserting a ProductionFeature without its ProductionMethod as well as Form and associated Part	151
Figure 8.6: Lightweight view of application specific ontologies developed from the MCCO and the instantiated disc design and production views of the application specific ontologies.	154
Figure 8.7: Implementation of the application specific ontologies, developed using the MCCO, in IODE	155
Figure 8.8: The knowledge to be captured for the Rim.....	157
Figure 8.9: The query to link Disc Design and Production Features through the MCCO	159
Figure 8.10: Assertion of a Rim fact 'Rim1' with NeckWidth =6mm and GrooveAngle=30deg ...	160
Figure 8.11: Assertion of Rim fact 'Rim1' with NeckWidth =11mm and GrooveAngle =60 deg..	161
Figure 8.12: Overview of the Meta level production methods being captured	163
Figure 8.13: Acquisition of StandAloneDiscFamilyProductionmethod from the XKS	164
Figure 8.14: Acquisition of RimProductionmethod from the XKS	164
Figure 8.15: Query for determining the accommodation of a FeatureProductionmethod in a PartFamilyProductionMethod	165
Figure 8.16: To find out the Manufacturability of Stages in Operations	166
Figure 8.17: The knowledge to be captured for the Rim.....	167
Figure 8.18: The query and its results to establish the route to knowledge sharing	169
Figure 8.19: Assertion of a WebProfile fact Web1' with R = 262mm and r =50mm	169
Figure 8.20: Assertion of a WebProfile fact Web1' with R = 260mm and r = 50mm.....	170
Figure A2.1 Lightweight formalisation fo Manufacturing Resource Extension	240
Figure A2.2: Lightweight Formalisation of extension of Fixture core concept.....	241
Figure A2.3: Lightweight Formalisation of extension of MachineTool core concept	242
Figure A2.4: Lightweight formalisation of ManufacturingProcess (Feng and Song 2003; Todd 1994).....	244

Table of Tables

Table 2.1: Types of ontologies developed further from (Zhou and Dieng-Kuntz 2004)	23
Table 2.2: Relevant Classification of Ontologies.....	23
Table 2.3: Different ontology development languages	33
Table 2.4: Relevant ontology development tools.....	36
Table 2.5: List of ontologies shortlisted from the reviewed ontologies.....	38
Table 3.1: Design Feature functionality matrix	60
Table 3.2: Matrix for Production Features vs their production method.....	63
Table 8.1: Results of assertions for various Feature concepts.....	151
Table 8.2: The results of assertion for application specific ontologies.....	156
Table 8.3: The relations and function for capturing Rim's production knowledge.....	158
Table 8.4: The relations and function for capturing Rim's production knowledge.....	168

Abbreviations

ADACOR	ADaptive holonic COntrol aRchitecture
AIF	ATHENA Interoperability Framework
AP	Application Protocol
API	Application Programming Interface
ATHENA	Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications
BFO	Basic Formal Ontology
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
CAX	Computer Aided Technologies
CIM	Computer Independent Model
CIMOSA	Computer Integrated Manufacturing Open System Architecture
CL	Common Logic
CLIF	Common Logic Interchange Format
CPM	Core Product Model
DIFF	Domain Independent Form Feature
DL	Description Logic
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
eCOIN	extended COntext INterchange
ERP	Enterprise Resource Planning
FCA	Formal Concept Analysis
FOL	First Order Logic
IC	Integrity Constraint
ICT	Information and Communications Technology
IDEF	Icam DEFinition
IODE	Integrated Ontology Development Environment
ISO	International Organisation for Standardization
KB	Knowledge Base
KBE	Knowledge Based Engineering
KFL	Knowledge Frame Language
KIF	Knowledge Interchange Format
KM	Knowledge Management
KMS	Knowledge Management Systems

LCKD	Life Cycle Knowledge Desktop
GD & T	Geometric Dimensioning and Tolerancing
GPU	Graphical Processing Unit
MAFRA	ontology MApping FRAMework
MANDATE	MANufacturing management DATa interchange
MASON	MANufacturing Semantics ONtology
MCCO	Manufacturing Core Concepts Ontology
MDA	Model Driven Architecture
MDI	Model Driven Interoperability
MLO	Mid Level Ontology
MRP-I	Material Resource Planning
MRP-II	Manufacturing Resource Planning
MSE	Manufacturing System Engineering
OCHRE	Object-Centred High-level REference ontology
OMS	Object Management System
OWL	Web Ontology Language
PDM	Product Data Management
PDM	Platform Description Model
PERA	Purdue Enterprise Reference Architecture
PFEM	Product Family Evolution Model
PIM	Platform Independent Model
PLIB	Parts LIBrary
PLCS	Product Life Cycle Support
PLM	Product Lifecycle Management
PSL	Process Specification Language
PSM	Platform Specific Model
PSRL	Product Semantic Representation Language
RDF	Resource Description Framework
RDF(S)	Resource Description Framework Schema
RM-ODP	Reference Model of Open Distributed Processing
SAMULET	Strategic Affordable Manufacturing in the UK with Leading Environmental Technology
CL	Simple Common Logic
SMIF	Semantic Manufacturing Interoperability Framework
STEP	STandard for the Exchange of Product model data
SUMO	Suggested Upper Merged Ontology
SUO	Standard Upper Ontology

SWRL	Semantic Web Rule Language
UI	User Interface
ULO	Upper Level Ontology
UML	Unified Modelling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
w.r.t	With respect to
WSDL	Web Service Description Language
XKS	eXtensible Knowledge Server
XML	eXtensible Markup Language

1 Introduction

1.1 Research Context

In an increasingly competitive market, industries need to provide better quality products, at cheaper rates and in a shorter amount of time (Goffin and Koners 2011; Huang et al. 2009) and knowledge sharing is one of the key factors in making it possible (Arthur and Huntley 2005; Collins and Smith 2006; Cummings 2004; Hansen 2002; Lin 2007; Mesmer-Magnus and DeChurch 2009).

When knowledge is being shared between humans, then the differences in understandings can be resolved through personal interactions but this is not possible when sharing knowledge between Information and Communication Technology (ICT) systems. Modern manufacturing organisations use multiple systems and therefore, identifying effective routes to share knowledge between and within these systems is important.

This leads to the importance of “interoperability” which is “the ability of two or more systems or components to seamlessly exchange information and to use the information that has been exchanged” (IEEE-Std-Computer-Dictionary 1991; ISO/IEC-TR-10000-3 1998). Interoperability problems have been estimated to cost the U.S. automotive sector \$1 billion annually (Brunnermeier and Martin 1999) and the U.S. capital facilities industry \$15.8 billion annually (Gallaher et al. 2009). Up to 70% of interoperability project costs are spent on identifying and reconciling the mismatches in semantics mismatches (Bussler et al. 2005). Semantic mismatches define the differences in the meanings of concepts.

Semantic mismatches can be reconciled and reduced through the use of common semantics (Hakimpour 2003) and ontologies are one of the ways of providing a common semantic base (Sánchez et al. 2007). Although the definition of ontology by Gruber (1995) is regularly quoted in literature, perhaps the definition of ontology given in ISO-18629 is easier to understand. According to ISO-18629 ontology is “a Lexicon of the specialized terminology along with some specifications of the meanings of the terms involved (ISO-18629-1 2004)”. This definition also encompasses the broader concepts of lightweight ontologies (taxonomies or hierarchies of terms) as well as heavyweight ontologies (formal or computer understandable) through the phrase “some specification of the meanings”.

Lightweight ontologies (defined in detail in section 2.4.2.1) consist of taxonomies or hierarchies of terms which make the terms in lightweight ontologies open to multiple

and possibly inappropriate interpretations (Young et al. 2007). Heavyweight ontologies (defined in detail in section 2.4.2.1) are formal (computer interpretable) ontologies built using logical theories, which are rigorously formalized (Uschold and Gruninger 2004). Heavyweight ontologies can help to formalize semantics of concepts and surmount the limitations of lightweight ontologies. Moreover, they offer better reasoning and inferring capabilities. Thus, they can provide a rigorous common semantic base, which can potentially offer a route to knowledge sharing between manufacturing domains.

Manufacturing knowledge has partially been managed using ERP, MRP, PLM, KMS, software tools and approaches. These applications have been limited in their ability to provide an environment for sharing knowledge (Abramovici and Sieg 2002) partly because of an underlying structure based on lightweight ontologies. In this thesis a heavyweight manufacturing ontology as a common semantic base is proposed to support knowledge sharing between production and product design.

1.2 An Overview of the proposed approach

An overview of the research approach is shown in figure 1.1. Even though the manufacturing industry deals with design, production, operation and disposal domains in the lifecycle of parts, only design and production domains are depicted in figure 1.1 because they are the focus in this thesis.

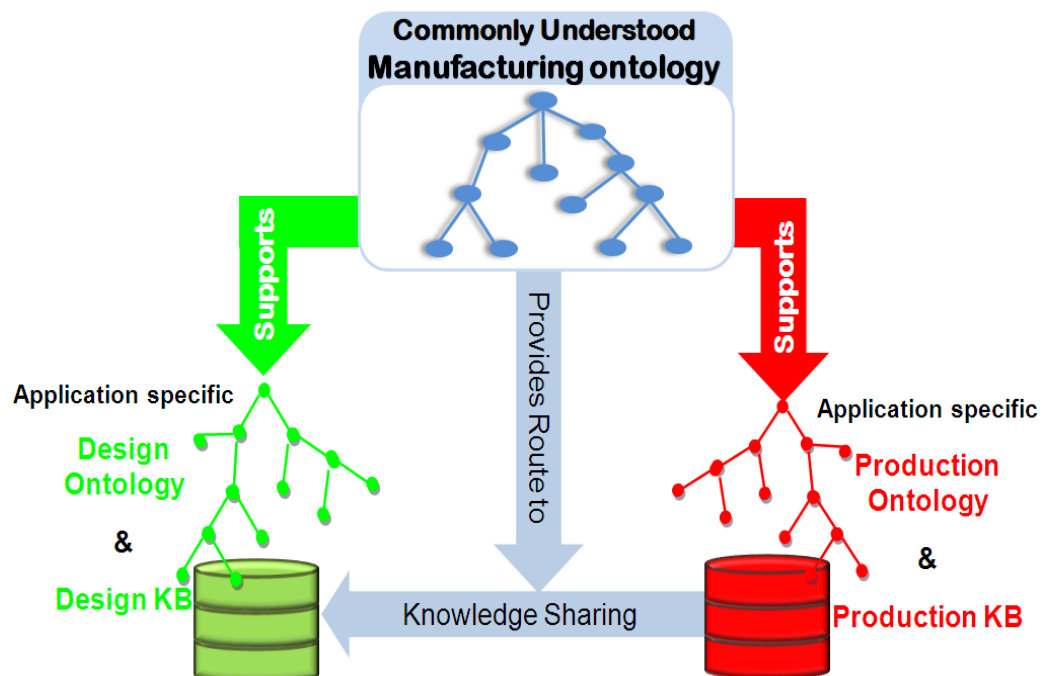


Figure 1.1: An overview of the research framework

The figure shows a commonly understood manufacturing ontology underlying these domains. The figure shows that this ontology supports the development of application specific product design and production ontologies and knowledge bases. The figure

illustrated that the manufacturing ontology should provide a route to relate product design and production domains to enable knowledge sharing across them.

1.3 Aims and Objectives

The aim of the research work reported in this thesis is to make a contribution to the understanding of the use of heavyweight ontological approaches to support effective capture and sharing of production knowledge with product design.

The achievement of this aim should contribute to the understanding of the use and exploitation of ontology based knowledge and decision support systems in PLM. It is proposed in this research work that meeting this aim requires the development of a manufacturing ontology, which supports the development of application specific product design and production ontologies thereby, provides a route to relate product design and production domains for knowledge sharing.

In order to meet the above aim a number of research objectives have been set. They are described below:

1. To identify a set of core manufacturing concepts and relations in the context of sharing production knowledge with product design;
2. To formalize the concepts and relations in the form of a lightweight ontology;
3. To formalize the semantics of identified core concepts and relations in heavyweight logic so that the interpretations of their meaning are unambiguous and consistent. Thus, enable knowledge system to identify similarities and differences between product design and production concepts;
4. To unambiguously capture the variations in depths of meaning of concepts from generic to very specific levels;
5. To use the identified and formally defined set of concepts to support the development of semantically sound application specific ontologies for product design and production domains.
6. To identify a way of relating product design and production domains through the formalised core concepts and relations;
7. To define concepts and relations and an approach to support the capture of and reasoning about the abstract production knowledge.
8. To design and conduct experiments in order to investigate and evaluate the above objectives and the success of the approach.

The above objectives give a high level generic view of the research work reported in this thesis. The details of the main research issues and the novel aspects of this research are explained in chapter 4.

1.4 Research Methodology

The research methodology to meet the above aims and objectives is that of hypothesize and test. The hypothesis for the thesis is given below.

1.4.1 Hypothesis

The hypothesis of the research work is stated as:

“An ontology of a comprehensive set of core manufacturing concepts defined in formal logic can support knowledge sharing across product design and production domains by providing a verifiable semantic base.”

The path of verifying this hypothesis includes the development of a high-level ontology of a comprehensive set of manufacturing core concepts formalized in heavyweight logic to capture unambiguous definitions of concepts. The high level nature of the ontology can establish its suitability as a reference ontology to support the capture of product design and production domain concepts. A commitment to this ontology can ensure the consistency of captured knowledge and can facilitate knowledge sharing between product design and production.

1.4.2 Ontology Development Methodology

The research reported in this thesis follows a manual ontology development methodology. The approach has been adopted because the ontology being explored in this research is a relatively high level one, and the manual approach offers a more rigorous and a more comprehensive structure at higher levels (Blomqvist and Öhgren 2008) as compared to the automatic approach. A manual approach also helps to identify and better define the essential concepts as the ontology becomes more specific (Blomqvist and Öhgren 2008). The ontology development methodology being used in this work is based on the guidelines provided by Blomqvist and Öhgren (2008) and Noy and McGuinness (2000). Figure 1.2 and the following text illustrates the ontology development methodology;

1. Understand the Problem: This is to be done through a review of the relevant literature and an industrial study to understand the latest research trends and explore the problems in developing a manufacturing ontology.
2. Requirements Analysis:

- a. Outline, the purpose, scope, uses and users of the ontology;
- b. Identify a set of key issues based on the industrial study and review of research work done by other researchers;
- c. Clearly outline the issues in the development of a manufacturing ontology for sharing production knowledge into product design;
- d. Explicitly list the requirements of the manufacturing ontology;

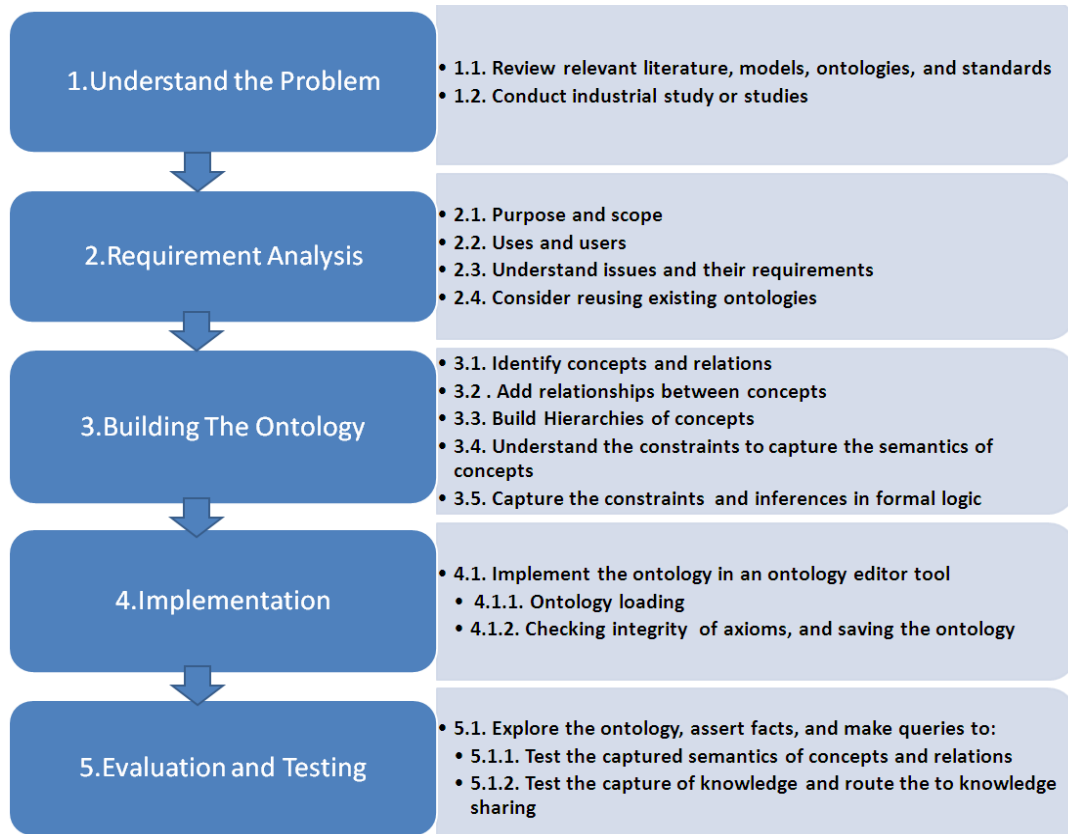


Figure 1.2: Ontology Development Methodology

3. Building the Ontology:
 - a) Identify the main categories of concepts;
 - b) Rationalize the understanding gained from different manufacturing ontologies to define a set of manufacturing concepts and relations;
 - c) Build a hierarchy of those concepts through simple relations e.g. the parent child relations;
 - d) Define the relations across different categories of concepts;
 - e) Understand the constraints on concepts and define them in formal logic to make the ontology heavyweight and capture the semantics of concepts unambiguously.
4. Implementation: Implement the ontology in the ontology development environment.
5. Testing: is to be carried out by asserting facts in the knowledge base and making queries in general to verify the hypothesis and to investigate the following:

- a) The formal capture of the semantics of concepts;
- b) The success of the proposed approach in capturing varying depths of meaning of concepts;
- c) The success of the proposed approach to enable the capture and acquisition of knowledge at multiple levels of abstraction;
- d) Whether any modifications are needed to be made in the ontology.

1.5 Research Scope

This thesis is part of a larger research project, Interoperable Manufacturing Knowledge Systems (IMKS), which constrains the focus of this research work. A brief introduction of the IMKS project and the focused area of research work are given below.

1.5.1 IMKS Project

The IMKS project addresses the issues of interoperability across product design and production domains of the product lifecycle. The IMKS project aimed to develop a knowledge system in the form of an ICT tool which facilitates sharing knowledge across product design and production domains. The IMKS idea is shown in the figure 1.3. As shown in the figure the IMKS system provides a common library of

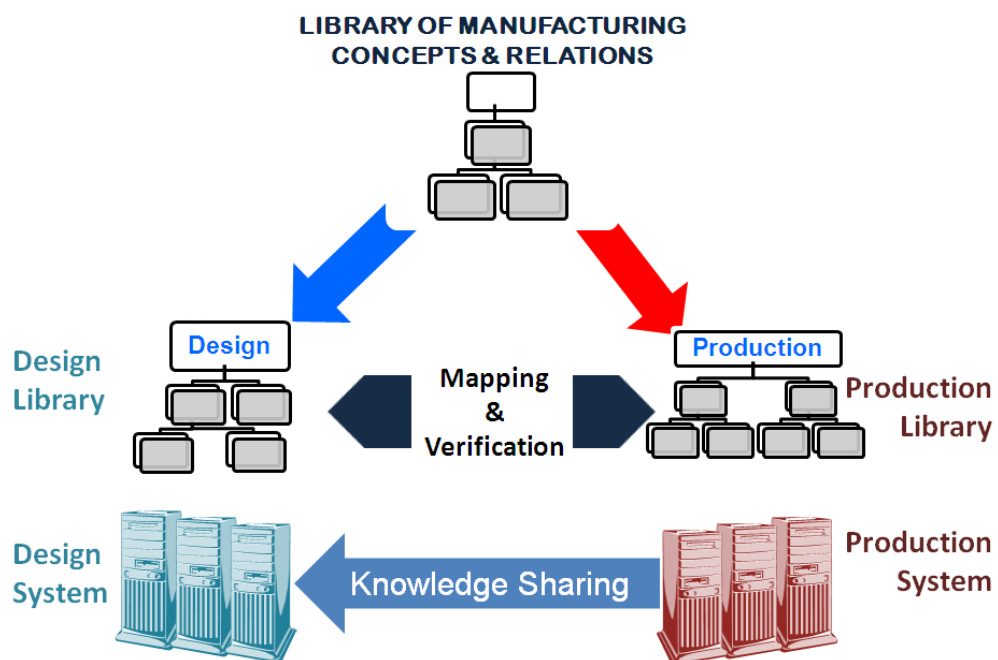


Figure 1.3: The IMKS concept

manufacturing concepts and relations. This library then support the development of libraries for design and production systems and knowledge sharing across those systems. There are two main research aspects of the IMKS.

The first aspect involves the development of a library of shared manufacturing concepts and relations and the formalisation of their semantics to support interoperability across product design and production domains. The research work reported in this thesis is focused on the development of that library of manufacturing concepts and relations.

The second aspect is to develop a verification mechanism to support the mapping of product design and production concepts. Work on this aspect has been conducted by Anjum (Anjum 2011).

The development of an integrated PLM and KM environment based on the results from the research has also been undertaken by the IMKS project team. An extension of this project may result in a system, which allows the manufacturing organization to share knowledge across various product lifecycle activities both within and across organizations. Further details about the project can be accessed from the project website at <http://lupo.lboro.ac.uk/research/product-realisation/imks/index.htm>.

1.5.2 Ontology Development Languages and Tools

The ontology development languages refer to the formal languages being used to formalise the ontologies and the ontology development tools refers to the software applications being used to implement the ontologies.

1.5.2.1 Ontology Development Languages

The IMKS project used the Unified Modelling Language-2 (UML-2) for the lightweight formalization of ontologies. For the heavyweight formalization a Common Logic based language i.e. Knowledge Frame Language (KFL) is used.

Throughout the thesis, UML-2 has been used to aid the ontology development process and provide a lightweight formalisation of the ontology. Knowledge Frame Language (KFL) is a Common Logic-based ontological formalism, developed by Highfleet Inc. that provides the syntax and first order semantics required for developing heavyweight ontologies. The ability to encode ontological content in KFL derives from Highfleet's Upper Level Ontology (ULO). An introduction to the use of KFL is provided in appendix A1.

1.5.2.2 Ontology Development Tools

The software applications being used to implement the ontology are Enterprise Architect V8.1, Notepad ++ and Integrated Ontology Development Environment v4.1. Enterprise Architect is being used for developing the UML-2 diagrams as a lightweight representation of the ontology. Enterprise Architect facilitates the modelling of ternary

and higher-arity relations and the modelling of powertypes and clabjects. The powertypes and clabjects are detailed in chapter 7.

For the heavyweight formalisation, coding of the ontology in KFL is done using Notepad ++ (Notepad++ Website, 2012). Notepad ++ is an open source software tool that facilitates coding programs in various file formats.

The ontology coded in KFL is implemented in the IODE. The IODE, developed by Highfleet Inc. (Highfleet Inc., 2012), is an ontological environment that is capable of handling heavyweight Common Logic-based ontologies and KBs. IODE constitutes the primary environment for the deployment and the experimental verification of the ontology. The ontologies are loaded into IODE as knowledge bases that are called 'Extensible Knowledge Server (XKS)'. An XKS holds the ontology and also acts as a knowledge base. In contrast to other ontology development tools like Protégé (Protégé Website, 2011), IODE uses text files that are coded in KFL.

1.5.3 Focused Production Area

A world leading aero engine manufacturing company is one of the main industrial collaborators in this research. Therefore, the industrial exploration of the research was focused on the design and production of an aero engine part. The production domain is further narrowed down to conventional machining and turning in particular.

Therefore, the concepts for the production domain are explored with the main focus placed on capturing conventional machining knowledge of an aero engine part and sharing this with product design. The important design concepts, which help to link the two domains and thus provide a route to interoperability, are also explored.

1.6 Thesis Structure

The main structure of the thesis is illustrated in figure 1.4 which is organized as follows. After the introductory chapter i.e. chapter 1, a state of the art review of ontology based manufacturing knowledge systems is presented in chapter 2. Chapter 3 presents an industrial investigation with one of the industrial collaborators to refine the understanding from an industrial perspective. Chapter 4 presents the novel aspects of this thesis and discusses the requirements of proposed manufacturing core concepts ontology. The next three chapters i.e. chapters 5, 6 and 7 propose the solutions to research problems and explain the three novel aspects of the research work in detail. Chapter 8 provides an experimental investigation of the solutions proposed in chapters 5, 6 and 7. Chapter 9 reports a discussion on the developed manufacturing ontology and presents the conclusions drawn and provides a guideline for future research.

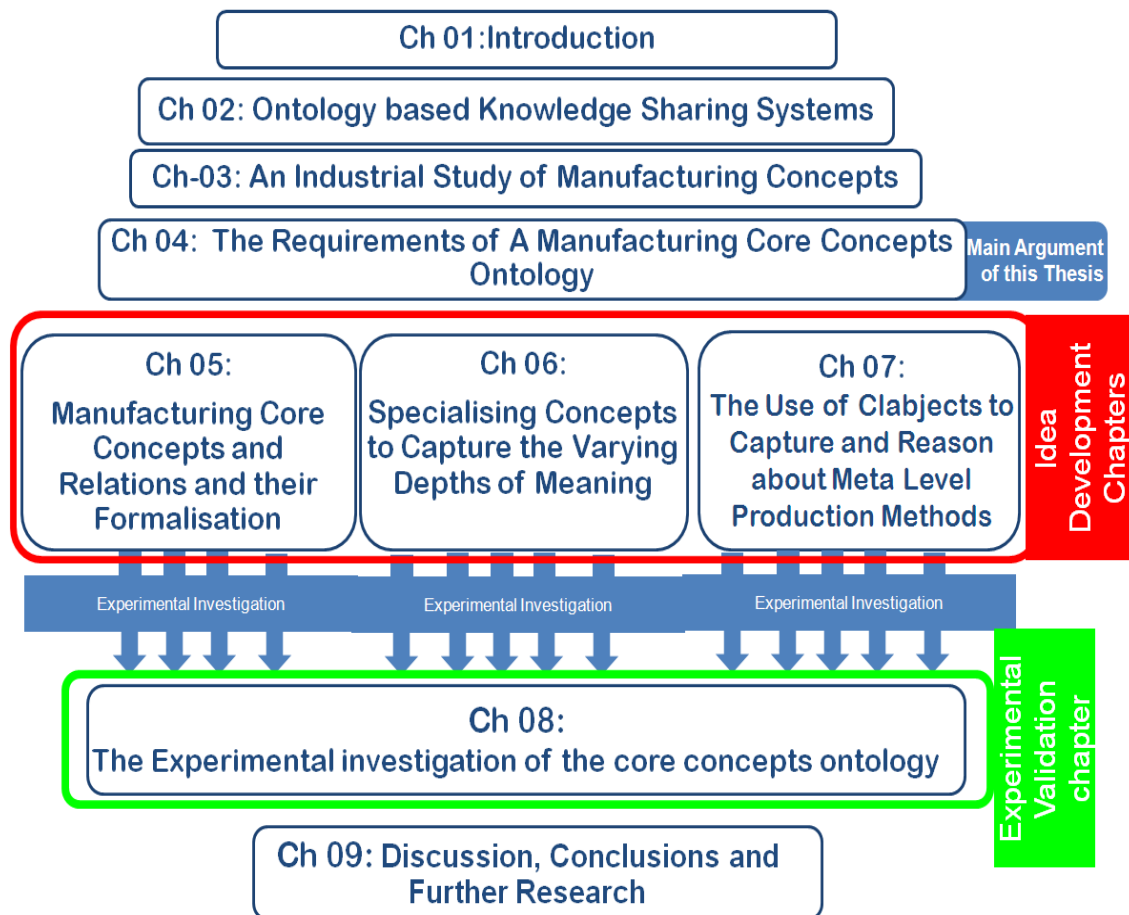


Figure 1.4: Thesis structure

2 Ontology based knowledge sharing systems

2.1 Introduction

This chapter presents a review of the state of the art in the field of ontology based knowledge sharing systems. Section 2.2 states the basic definitions of data, information and knowledge. Section 2.2 also highlights the issues in achieving knowledge sharing and interoperability in manufacturing. Section 2.3 describes different methods of knowledge sharing. Section 2.4 illustrates the ontological methods for interoperable systems. This section illustrates the definitions and types of ontologies. The section also illustrates different ontology development methodologies, languages and tools. Section 2.5 presents a review of state of the art ontologies and ontological approaches used in manufacturing. Altogether, sections 2.2 through to 2.5 enable the identification of key research issues which are summarised in section 2.6.

2.2 A General View of Knowledge Sharing and Interoperability

In order to develop an understanding of knowledge sharing and interoperability and the issues related to it, it is important to clearly define the term 'knowledge'. The definition of knowledge is not easy as it is different according to different authors. The terms 'Data' and 'Information' are often used along with the term 'Knowledge' (Wiig, 1994; Sveiby, 1997; Spek and Spijkervet, 2005). Therefore, to understand what 'Knowledge' means, the terms 'Data' and 'Information' need to be understood.

2.2.1 Definitions of Data, Information and Knowledge

The interpretations of these terms vary according to their field of interest. Because this thesis is broadly within the area of application of ICT to manufacturing knowledge sharing, therefore, data, information and knowledge should be defined in that context. There are various reported definitions of 'data'. Some defined 'data' as the alphanumeric and symbols arranged without any order which do not generate any meanings (Spek and Spijkervet 2005) while others, defined Data as simple observations of the states of world (Davenport et al. 2000). The definition of data as "the alpha-numeric and symbols arranged without any specified order which may or may not generate meaning for machines or humans" is considered appropriate because it captures the machine as well as human understanding of data. Even if data generates some meanings, it is not intended to have any.

Like Data, information has also been defined differently by different researchers. Some defined information to be meaningless (Nonaka and Takeuchi 1995; Sveiby 1997), whereas, most defined it to have meanings (Davenport et al. 2000; Spek and

Spijkervet 2005; Wiig 1994). The definition of Information as “Information exists when the relations between data are recognized within a specific context” (Cochrane et al. 2008) is considered relevant for this thesis. This is because it provides an ontological aspect by considering the context of Data and this generates meanings. For example, entries like 35, >, 40, machine1, 90um are data. However, this data becomes information when put into a context e.g. 40 > 35, surface finish of machine1 is 90um.

As information is more meaningful than Data, similarly knowledge is more meaningful than information. Like data and information several definitions of knowledge can also be found in literature. Some defined knowledge as the useful information (Davenport, 1997), some defined it as the human intellect added to information (Spek and Spijkervet, 2005). Knowledge has also been defined as information plus the additional details about the use and application of that information (Harding 1996) and knowledge is composed of concepts (Sánchez et al. 2007).

In the context of this thesis, the definitions given by Harding (1996), Sanchez et al (2007) and Cochrane et al (2008) are found useful. According to these definitions, Knowledge is made up of concepts that provide ‘information’ about the ‘information’. where the later ‘information’ compliments the first one by describing its use and the ways and rules of its use. Thus, knowledge is constituted of concepts with some rules that describe the actions to be taken when certain information is there (Cochrane et al. 2008). This definition of knowledge is suitable because it captures the understanding that knowledge is more explicit, meaningful and useful than information and also because this definition is in line with the notion of heavyweight ontologies (which are explained in section 2.4.2.1.2).

2.2.2 Levels of Knowledge

Knowledge can exist at different levels with respect to different degrees of abstraction. The levels of knowledge that are relevant for this thesis are defined by Turban and Arons (2005) as deep level knowledge and shallow level knowledge.

2.2.2.1 Shallow Knowledge

This refers to the surface level knowledge (actual declared or asserted facts in a KB) which is very specific. This knowledge only consists of the facts asserted into a knowledge base and is based on an underneath structure. Examples, of shallow knowledge can be assertions like “MachineTool1 can machine part 1”.

2.2.2.2 Deep Knowledge

This is concerned with the underneath structure of knowledge which captures the logic and causal relation working behind the system. This also keeps in consideration the possible relations between system modules. This knowledge is applicable to multiple situations and scenarios." It is much more difficult to make computational use of this knowledge as compared to shallow knowledge (Turban and Arons, 2005). Examples of deep knowledge can be the concepts like *ProcessPlan*, *CuttingTool* and *MachineTool* as well as their relations that are used to instantiate the actual detailed shallow level process plans.

2.2.3 Types of Knowledge

Knowledge can be categorised into different types with respect to its explicitness and configuration.

2.2.3.1 Types of Knowledge with respect to Explicitness

Zhou and Dieng-Kuntz (2004) summarized the composition of manufacturing knowledge into three types with respect to the explicitness i.e. explicit, implicit and potential. Where explicit knowledge is capture-able on papers or in systems, implicit knowledge is in the heads e.g. skills and expertise and potential knowledge is the knowledge that has the potential to be illustrated explicitly but has not yet been explicitly defined.

2.2.3.2 Types of Knowledge with respect to Configuration

Turban and Aronson (2005) categorized knowledge into the three types; Declarative Knowledge, Procedural Knowledge and Meta Knowledge.

Declarative knowledge is made up of facts. For instance, the assertion like "Machine-1 can machine surface-1". This knowledge is based on the facts asserted in a knowledge base system. Declarative knowledge is always at the shallow level.

Procedural knowledge captures the procedure of carrying out a task or activity and the conditions under which that task is to be performed. For instance "A surface may be machined if its required surface finish is less than the capability of milling machine which is 100um" is procedural knowledge. Procedural knowledge in manufacturing is related to the capture of manufacturing methods and processes. Procedural knowledge can either be at deep or shallow levels of knowledge. If detailed procedural knowledge is captured which contains actual instances then it is shallow. If only the abstractions of actual procedural knowledge are captured then that would be at deep level.

Meta knowledge is the knowledge about knowledge. This captures the reasoning behind the declarative and procedural knowledge. This captures how a system actually makes inferences and decisions by capturing the logical arguments behind a system. This knowledge is at the deep level of knowledge.

2.2.4 Manufacturing Knowledge

Manufacturing is and will be one of the top revenue and employment generators in Europe. Manufacturing is responsible for nearly 22% of the EU GNP and manufacturing relates to , about 75% of total GDP and 70% of the employment in the European Union (EU) (Manufacture 2004). In 2005, 2.3 million enterprises in the EU-27 had manufacturing as the main activity, having generated EUR 6,323 billion turnover, value added production of 1,630 billion and having employed 34.6 million of human resource (Manufacture 2004; Trade-policy-review-report 2009).

Manufacturing knowledge according to (Zhou and Dieng-Kuntz 2004) is the collection of facts and data that the manufacturing industry require to define the set of activities that implement production. Manufacturing knowledge can belong to the process planning, product design, assembly, operations and services, and disposal. It can be the product model knowledge e.g. the Core Product Model (CPM) (Fenves et al. 2006), Product Data knowledge models e.g. STEP (ISO-10303-1 1994), part libraries e.g. PLib (ISO-13584 2001) etc.

2.2.5 Knowledge sharing as an Issue in Manufacturing

Knowledge sharing is the most important area of knowledge management (deals with capturing, storing, reusing and sharing knowledge (Lee 2001)) and plays a key role in effective knowledge management (Hendriks 1999). Knowledge sharing is one of the key factors in bringing down the cost of production, time to market in new product development. Knowledge sharing is also a key factor for making the performance of products/services meet or exceed customer's expectations (Arthur and Huntley 2005; Collins and Smith 2006; Cummings 2004; Hansen 2002; Lin 2007; Mesmer-Magnus and DeChurch 2009).

Sharing of production knowledge with product designers is of key importance because in manufacturing industry, design is 5% of the total industrial activity which affects up to 70% of the total cost (True and Izzi 2002) as shown in figure 2.1. Designers use 30%-70% of their personal knowledge (Court 1998). Designers spend more than 70% of time searching and handling recently updated knowledge (Kuffner and Ullman

1991) because of the problems in knowledge organization and sharing (Lee et al. 2005).

Problems in knowledge sharing cost the Fortune 500 companies a minimum of US\$ 31.5 billion annually (Babcock May-2004). As such, the productivity of designers can potentially be improved through effective knowledge sharing and reuse.

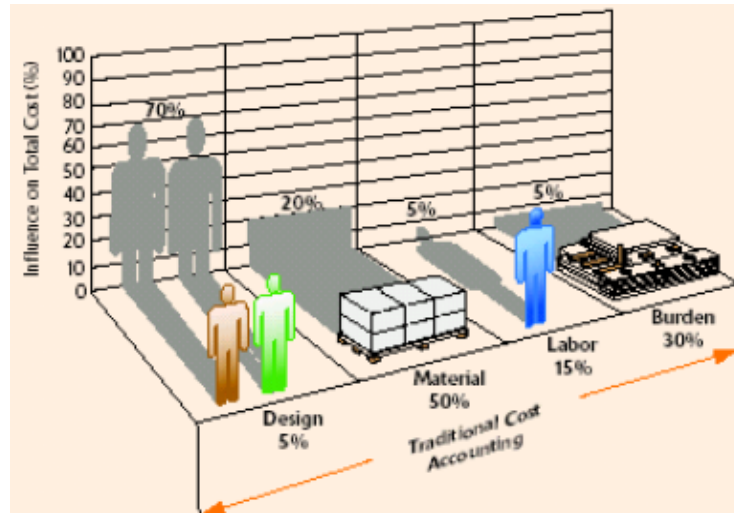


Figure 2.1: The influence of design on costs (True and Izzi 2002)

In order to share knowledge more effectively, a mechanism is required to communicate across different domains. In an ICT context, knowledge sharing across software modules and system components, leads to the concept of interoperability, which is a requirement to enable knowledge sharing.

2.2.6 Interoperability

Interoperability may be defined as “The ability to share technical and business data, information and knowledge seamlessly across two or more software tools or application systems in an error free manner with minimal manual interventions”(Ray and Jones 2003). However, the definition of interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” (IEEE-Std-Computer-Dictionary 1991; ISO/IEC-TR-10000-3 1998) is relevant to this thesis because it covers the interaction across systems as well across system components. In this context, manufacturing interoperability is related to the ability to share technical & business information between different departments in a factory plant or between the organisations in an extended manufacturing enterprise (Borgo and Leitão 2007).

As mentioned in section 1.1 of chapter 1, billions of US\$ are spent annually on solving interoperability problems in the US automotive sector only. If those billions of

US\$ are scaled to include other manufacturing sectors like aeronautical, textiles, processing industry, electronics, and furniture and across not only the USA but the whole world, the figures would increase substantially. Therefore, interoperability problems represent a major issue and it is imperative to minimise the billions of dollars spent on solving this issue and finding a solution that enables better interoperability.

Interoperability can be broadly categorised into syntactic interoperability, semantic interoperability and community interoperability (Briefing paper 2008). Syntactic interoperability is “the ability of systems to process a syntax string and recognise it as an identifier even if more than one such syntax occurs in the systems”. Semantic interoperability is the ability of systems to determine if two concepts refer to the same meanings; and if not, how the two concepts are related. Community interoperability “is the ability of systems to collaborate and communicate using identifiers whilst respecting any rights and restrictions on usage of data associated with those identifiers in the systems”.

This thesis is mainly targeted at achieving the semantic interoperability and knowledge sharing between product design and production domains.

2.3 Methods to Achieve Interoperability

There have been several ways of sharing the knowledge both within and across organizations. The simplest and oldest method is from person to person. This involves knowledge sharing through verbal discussion, and personal meetings, through e-mails and other means of communication between persons. However, these approaches make the knowledge sharing dependant on the availability and abilities of the personnel. This approach can result in loss of precious industrial time and in business time is money. Therefore, the approaches involving the use of ICT to share knowledge are explored. A review of these architectures is provided by organising them as the model driven interoperability and frameworks for interoperability.

2.3.1 Model Driven Interoperability (MDI)

The Model Driven Interoperability (MDI) is based on the systems development approach for developing multiple integrated systems known as Model Driven Architectures (MDA). The MDA idea was initiated by the Model Driven Software Development (MDSD) group, and is now one of the recommended specifications from the Object Management Group (OMG) (Bourey 2007).

The MDA methodology is composed of a set of fundamental concepts defined in the MDA Guide (2003). These include 1. Computation Independent Model (CIM), 2. Platform Independent Model (PIM) and 3. Platform Specific Model (PSM). The interaction between these models is basically transformation of CIM into PIMs and PIMs into PSMs as illustrated in figure 2.2.

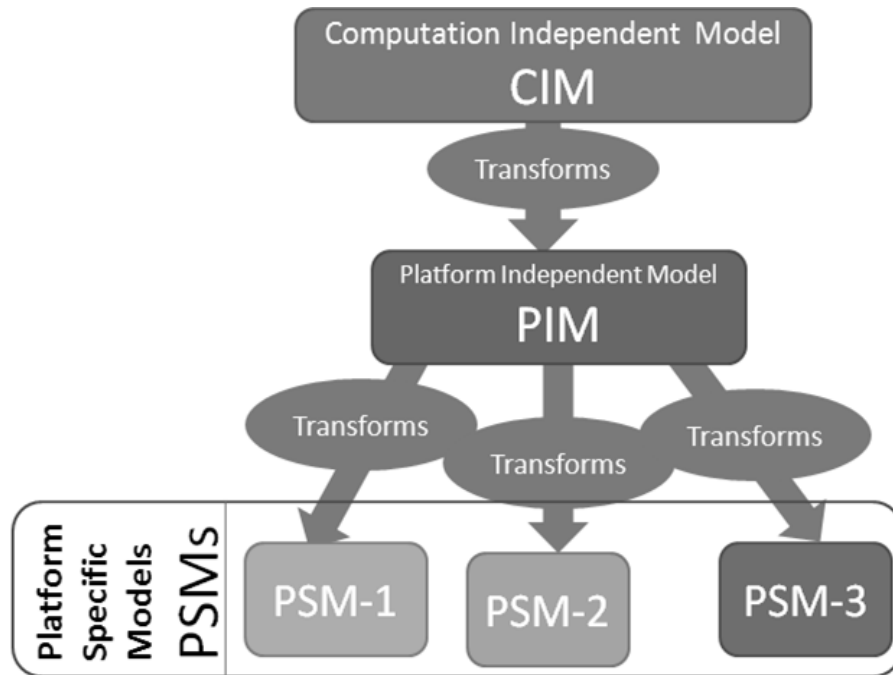


Figure 2.2: MDA approach to interoperability

The CIM illustrates the business context and requirements for the system concerned, related to a computation independent viewpoint (Elvesæter et al. 2006). The PIM, however, defines a model at a level where it is used to explain the computational solution from a software tool independent view (Bourey 2007). Using transformation mechanisms a single PIM can be converted into one or more PSMs as shown in figure 2.2. A PSM is developed with a platform specific viewpoint and describes the realisation of software systems in the selected execution platforms (Elvesæter et al. 2006). A PIM supports multiple PSMs. An example of this can be seen in the use of STEP (ISO-10303-1 1994) standards as PIM models supporting multiple CAD specific PSMs

The principle of the MDA approach to interoperability, i.e. Model Driven Interoperability (MDI), is of interest as many researchers have made use of this to solve specific research problems (Cutting-Decelle et al. 2006; Didonet del Fabro et al. 2008; Elvesæter et al. 2006; Moalla et al. 2008; Staub et al. 2008).

Figure 2.3, presents a simplistic adapted version from (Lemrabet et al. 2010) showing how MDI can be used to solve interoperability problems. It shows two enterprises having adopted the MDA approach for developing their systems. Model transformations are there from CIMs to PIMs to PSMs for each enterprise. The interoperability between the different MDA levels across the two enterprises is facilitated by the intermediate interoperability models that support the CIM, PIM and PSM levels.

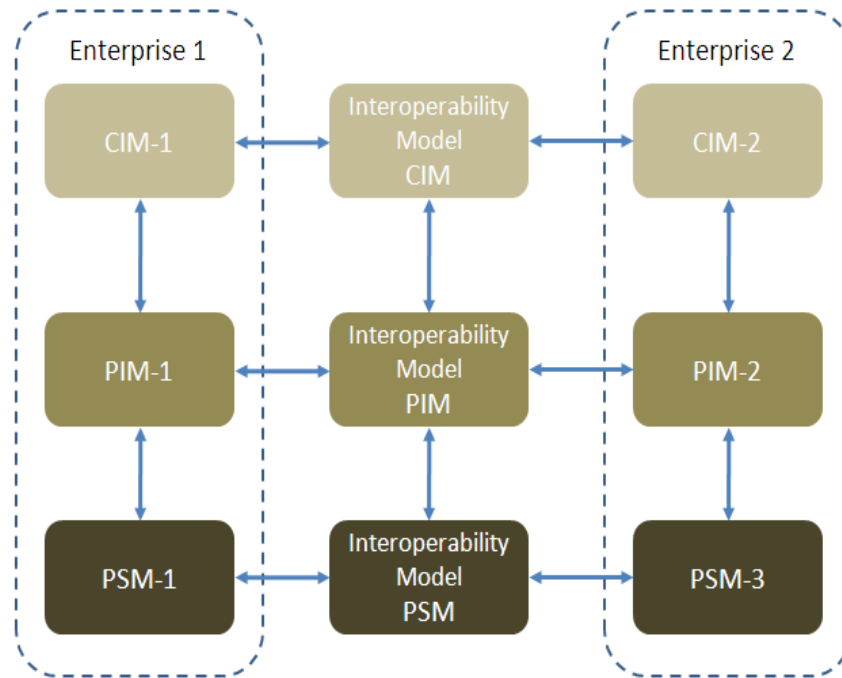


Figure 2.3: Reference Model for MDI adapted from (Lemrabet et al. 2010)

Present trends in MDI lead to the fact that MDA approach has been used to support interoperability (Grangel et al. 2007) as well as the capture of semantics. For instance, MDI was deployed to enhance the product data quality across the vaccine supply chain (Moalla et al. 2008). MDA was used to encourage semantic interoperability across Object-Oriented models (Staub et al. 2008). Bourey et al (2006), for example, refined the current knowledge models and transformations using the MDA and MDI. These models have been applied to the test cases within the INTEROP NoE project (Panetto et al, 2004).

MDA has gained extensive influence as a way of conceptualising generic models from various specific problems. The main approach of MDA is to detach the conceptual matters from implementation-specific matters and then compile them into one executable system (Oberle 2004). From the breadth of work performed in the field of MDA and MDI, it becomes obvious that there is an acknowledged importance of these approaches to interoperability and semantics.

MDA and MDI, however, do not completely address all requirements for interoperability and semantics. Firstly, it requires a compilation which prevents modification at run time. Secondly, it is not possible to query, infer, or reason about an MDA itself. Hence, it does not provide a way to query about the system structure and system components. However, Ontology-based approaches can potentially address these issues.

2.3.2 Frameworks for interoperability

As the understanding of interoperability grew, different interoperability frameworks were established to meet the requirements of interoperability at technical as well as business levels. Different interoperability frameworks like the Zachman Framework (The Zachman Framework Website, 2009), IDEAS Interoperability Framework (IIF) (IDEAS, 2003), The Open Group Architecture Framework (TOGAF) (TOGAF Website, 2009), and the ATHENA interoperability framework (AIF), identify multiple dimensions of interoperability. A brief review of the relevant frameworks is presented here.

2.3.2.1 IDEAS Interoperability Framework

In the IDEAS interoperability framework, a specific dimension is acknowledged for the implications of semantics across the “business”, “knowledge” and “ICT Systems”. The IDEAS interoperability framework is illustrated in figure 2.4. The figure shows that the IDEAS interoperability framework is designed to support semantic interoperability across enterprises. The interoperability is supported at the business, knowledge, and ICT system levels through the integrated, unified and federated approaches (these approaches are detailed in section 2.3.2.3).

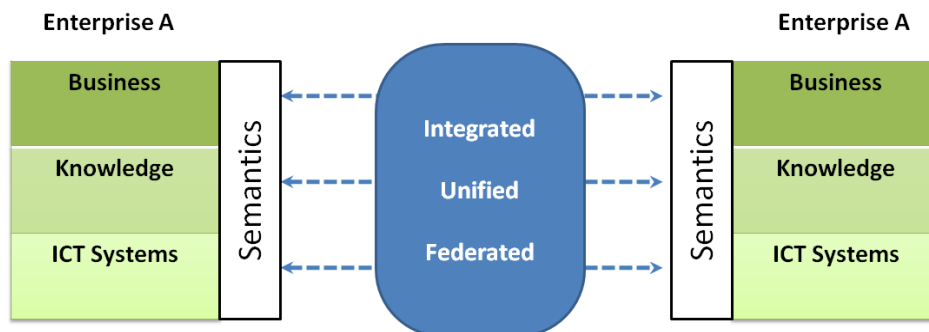


Figure 2.4 IDEAS interoperability framework redrawn from Chen et al (2004)

2.3.2.2 ATHENA Interoperability Framework (AIF)

The IDEAS interoperability framework was developed as part of the ATHENA (i.e. Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks & their Applications) project (Ruggaber, 2006). The IDEAS lead to the development of

ATHENA Interoperability Framework (AIF) (Berre et al. 2007). A simplistic view of the AIF is presented in figure 2.5 which shows that another dimension of 'service' has been added to the IIF and the 'ICT systems' dimension has been replaced with 'information/data'. At the business levels the AIF supports collaborative enterprise modelling whereas, at the processes level it supports business processes across organisations. At the services level the AIF supports flexible execution and composition of services and at the information/data levels it enables information interoperability. The AIF uses the MDI and ontologies to support semantic interoperability across enterprises as shown in figure 2.5. The AIF was designed to support interoperability not only across organisations but also within an organisation using the same framework.

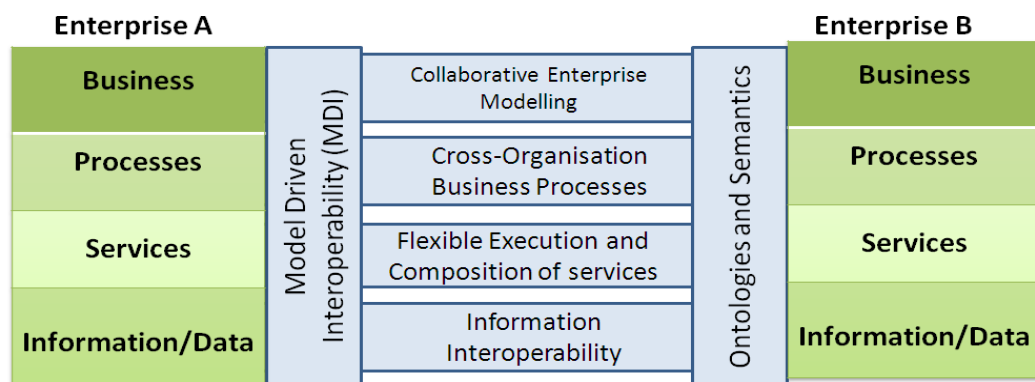


Figure 2.5: ATHENA interoperability framework (AIF) redrawn from Berre et al (2007)

2.3.2.3 Standard on Frameworks for Interoperability

The AIF laid the foundations for the development of the standard on Frameworks for Enterprise Interoperability (ISO/CEN-11354 2008). This standard proposed a three dimensional framework for enterprise interoperability which is illustrated in figure 2.6.

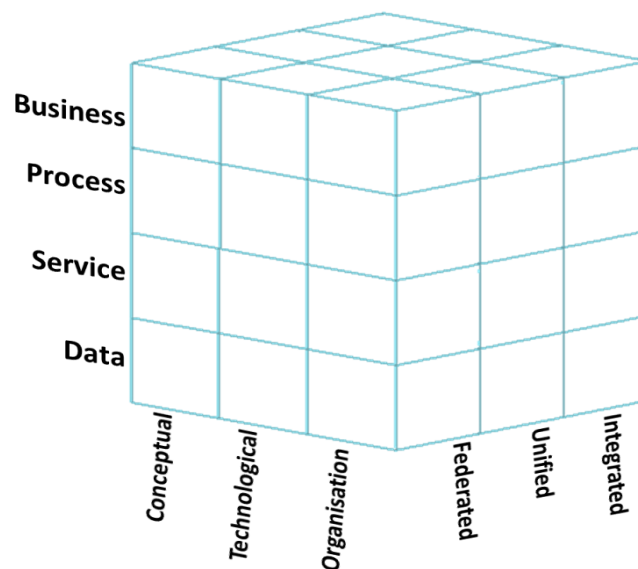


Figure 2.6: Frameworks for enterprise interoperability (ISO/CEN-11354 2008)

The three dimensions of the framework are (1) the interoperability barriers: which are conceptual, technological and organisational (2) interoperability concerns which are business, process, service and data (3) interoperability approaches which are integrated, federated and unified. These approaches were first identified in the standard for concepts and rules for enterprise models (ISO 14258). The withdrawal of this standard is scheduled. These approaches are explained as follows.

2.3.2.3.1 Integrated Approach

The Integrated approach is aimed at the complete integration that results in interoperability. In this approach all the interoperating partners are interconnected through a detailed standard structure to reach a common goal. This means all the partners that are to be interoperated should have to use the same terms and syntax. The syntax and semantics should be such that they can be shared, transported, translated (Ozman 2006) and should be understandable by all the participants. Standard based approaches e.g. STandard for the Exchange of Product model data (STEP)(ISO-10303-1 1994) Parts library standard and ebXML are examples the integrated approach. A number of knowledge integration methods can be found in literature (Bless et al. 2008; Blomqvist and Öhgren 2008; Kwon et al. 2007; Ozman 2006).

The multi-agent system COSMOA (Bloodsworth and Greenwood 2005) that facilitated the process of making decisions at large scale incidents in the medical domain, is also an example of integrated approach. Bless proposed a general and heuristic scheme to mark and model the knowledge attributes required to locate the better and preferred knowledge, and integrate knowledge bases (Bless et al. 2008). Ozman worked on the agent technology to integrate knowledge bases (Ozman 2006).

Integrated approach is effective and useful for the design and implementation of new enterprise systems. However, this approach is not particularly effective for the reengineering of an existing enterprise system. This approach limits the flexibility of different participants by defining a strict standard. Moreover, this approach is mostly limited to intra organizational level because of the standardised common structure which may not suit the outside organisation for integration without completely changing their adopted systems.

2.3.2.3.2 Unified Approach

In this approach to interoperability, a common meta-level structure is developed as opposed to a common standard structure for the participants. Because the common structure is at the Meta level, it is not directly executable. However, it offers a

common platform or framework to which all the participants can map and build their specific knowledge domains while strictly following it. The commitment to the common Meta structure offers flexibility in modelling the domains and also supports interoperability across them. This approach is best suited for interoperability between collaborating enterprises, organizations, vendors/suppliers, or those which are networked together. It is also suitable for interoperability between diverse departments of a manufacturing organisation. Since the participants have to map to the common Meta level structure instead of having a common form, this approach requires lesser time and cost for interoperability.

There can be minor losses of information due to the generality of common structure. These losses can be attributed to minor in-coherence at the instance level between different interoperating partners.

The process specification language (PSL) (ISO-18629) is an example of unified approach where the PSL offers a common Meta structure for multiple process domains to interoperate.

2.3.2.3.3 Federated Approach

This approach, involves no standard structure or common Meta structure between the interoperating entities. In this approach a dynamically evolving Meta structure is present between the participants. The models of all the partners as well as the Meta model should adapt to the interoperability requirements with time and with the addition and removal of participants. No participant can impose its own Meta structure as a common structure in this approach.

In this approach, interoperability is achieved by providing information about the entities involved at the run time. Capability profiles of inputs and outputs for the participating entities (with their syntax and semantics) are developed. The corresponding input and output information is mapped across entities and inconsistencies are found and sorted manually to support interoperability.

The extent of interoperability achieved through this approach is the maximum. The federated approach may use the Meta models for mapping several entities. These Meta-models are not imposed by any one of the partners and are not pre-defined. Moreover, the Meta structure in federated approach has a dynamically evolving structure. The federated approach is more suited for peer to peer interoperability and for the virtual enterprises and organizations.

However, the practicality of the federated approach is very difficult and limited due to its dynamic nature.

A specific information and support for federated approach can be found in the entity profiles that point out specific entity characteristics and properties required for achieving Interoperability (ISO 15745 and ISO 16100). An example of federated approach is the work done for interoperability of product lifecycle knowledge by Chen et al (2009) which is discussed in detail in section 2.5.1.4.

2.4 Ontological Methods for Interoperable Systems

Ontology based methods are useful for effective fast paced knowledge management. This approach has shown promise in product design and production domains by providing the hierarchical structures that support the capture of commonly agreed knowledge (Chang et al. 2010). Ontological methods are good at the illustration, management, and rationalisation of the intricate relations amongst different domains and their concepts (Chang et al. 2010). Ontological methods occupy a space at the core of the software applications and systems that help to share knowledge (Benjamin et al. 2006).

2.4.1 Definition of ‘Ontology’

The term ‘Ontology’ is borrowed by computer sciences from the metaphysics branch of philosophy where it deals with the nature of being (Oxford Dictionary, 2012). In ICT the most quoted definition of ontology is “an explicit specification of a conceptualization” by (Gruber 1995) but several other definitions of ontology are found in literature. Borst (1997) slightly modified Gruber’s definition to “a formal specification of a shared conceptualization”. Studer et al (1998) combined the definitions given by Borst (1997) and Grubber (1993) as “an ontology is an explicit and formal specification of a conceptualization”. Several other definitions of ontology can be found in literature (Uschold and Gruninger, 1996; Guarino, 1997; Roche, 2000; Noy and McGuinness 2000; Blomqvist and Ohgren, 2008).

However, the definition of ontology as “a Lexicon of the specialized terminology along with some specifications of the meanings of the terms involved” (ISO-18629-1 2004) is perhaps easier to understand and is also found more relevant for the research work reported in this thesis. This is because it covers both the lightweight and heavyweight understanding of ontologies which are discussed during different types of ontologies in the section 2.4.2.

From the definitions reported in literature it is concluded that a typical ontology will consist of a finite list of terms and their relations (Antoniou, G. 2008). Five basic primitives of ontology are: classes or concepts, relations, functions, axioms and instances (Liping et al. 2007). The behaviour of concept is controlled partially through the relations and functions and mainly through constraints in the form of axioms.

2.4.2 Types of Ontologies

Several types of ontologies with respect to different criteria have been mentioned in literature. A summary of these types and the criteria for categorising the ontologies into those types is presented in table 2.1.

Table 2.1: Types of ontologies developed further from (Zhou and Dieng-Kuntz 2004)

Author	Criteria	Name
Mizoguchi	Reusing	Contents
	Sharing	Communication
	Retrieval	Indexing
	Representing	Meta
Ushol et al	Formality	Highly Informal
		Semi-Informal
		Semi Formal
		Rigorously Formal
Van Heijst et al	Amount and type of structure of conceptualization	Terminological
		Information
		Knowledge Modeling
	Subject of Conceptualization	Application Ontologies
		Domain Ontologies
		Generic Ontologies
Guarino	Generality	Representation Ontologies
		Top level
		Task
		Domain
Lessila et al	Formality	Application
	Formality	Formality
Gomez-Perez et al	Formality	Similar to Lessila et al
		Lightweight Ontologies
Ushold & Jasper	Subject	Heavyweight Ontologies
		Synthesis of Heijst's and Guarino's terminological Ontologies
Borgo & Leitao	Formality	Formal-Foundational Ontologies
		Formal-Core Ontologies
		Foundational
Gangemi & Borgo	specificity	Core
		Domain
		Domain

In the context of this thesis, the two main criteria for classification of ontologies are (1) the specificity or subject of an ontology and (2) the formality of conceptualisation. Table 2.2 presents the types of ontologies with respect to these criteria. The definition and discussion on these types are presented in section 2.4.2.1 and 2.4.2.2.

Table 2.2: Relevant Classification of Ontologies

Criteria	Types
Formalisation of conceptualization	Lightweight Ontologies
	Heavyweight Ontologies
Specificity and subject of conceptualization ²³	Foundational Ontologies
	Core Ontologies
	Domain Ontologies

2.4.2.1 Types of Ontologies with respect to Formalisation

2.4.2.1.1 Lightweight Ontologies

Lightweight ontologies are mainly based on simple taxonomies (Borgo and Leitão 2007) with simple parent child relations between concepts. Lightweight ontologies are usually taxonomies that are made up of a set of concepts and the hierarchical relations amongst them (Zhu and Madnick 2007). Examples of these are WordNet (WordNet 2010), International standards like different application protocols of ISO-STEP (ISO-10303), P-Lib (ISO-13584, 2001), etc are examples of lightweight ontologies that have either no or weak constraints on the concepts such that their semantics cannot be interpreted fully and correctly by systems (Dartigues et al, 2007). Lightweight ontologies can be used for offering a shared understanding between humans in an organised way however, they are not sufficient for effective interoperability across systems (Dartigues et al, 2007).

2.4.2.1.2 Heavyweight Ontologies

Formal or heavyweight ontologies are based on rich logic (Borgo and Leitão 2007) which provide restrictions on semantics of concepts and model them rigorously (Gómez-Pérez et al. 2004). Formal ontologies use the axioms to explicitly capture and represent the semantics of concepts and thus they provide the inference capability (Zhu and Madnick 2007).

Formal ontologies have the capability to support the interoperability between multiple ontologies by providing a means of formally interpreting the meanings of concepts (Dartigues et al 2007; Gunendran and Young 2007; Chungoora 2010). An important advantage of heavyweight ontologies over lightweight ontologies is their ability to formally capture the semantics of concepts and their mappings which are explicitly represented and captured in formal mathematical logic (Zhu and Madnick 2007).

2.4.2.2 Types of Ontologies with respect to the Specificity

With respect to the specificity of conceptualisation ontologies can be categorised into (1) foundation ontologies (to cover the semantics of everything) (2) core ontologies (to cover the semantics shared across multiple domains) (3) domain ontologies (to cover the semantics for a specific domain only). More details about these types of ontologies are given as follows.

2.4.2.2.1 Foundation ontologies

The foundational ontologies i.e. the ontologies developed independent of any particular domain with a view to cover the semantics of everything, can provide a

common semantic base for any domain. Heavyweight foundation ontologies e.g. Suggested Upper Merged Ontology (SUMO accessed 12-04-2011), Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Basic Formal Ontology (BFO), Object-Centred High-level Reference Ontology (OCHRE) (Masolo et al. 2003) and Upper Level Ontology (ULO) (IODE 2010) have formally defined sets of concepts.

The concepts in foundation ontologies tend to be very generic e.g. *Particular*, *Endurant*, *Perdurant*, etc. (from DOLCE) and *Abstract Entity*, *Concrete Entity*, *Object* and *Event* (from ULO). Because the conceptualization of foundation ontologies is aimed at providing a broad coverage for several domains, they become overly generic and wide-ranging (Borgo and Leitão 2007) for effective use in specific domains. Therefore, the common semantic base provided by foundation ontologies will be too generic to be used directly for effective interoperability across focused domains like product design domain and production domain. Therefore, ontologies which are relatively more focused than the foundation ontologies are required to support interoperability across specific domains. The foundation ontologies can still be used to provide the semantic base for such ontologies (Borgo and Leitão, 2007).

2.4.2.2.2 Core Ontologies

Core ontologies are relatively less researched in literature (Gangemi and Borgo 2004). Core ontologies are not as generic as foundational ontologies and not as specific as domain ontologies. They cover the gap between very generic foundational concepts and very specific domain concepts (Usman et al. 2011). An intermediate set of concepts and relations between foundational and specific domain ontologies may be referred to as a core ontology or a core concepts ontology (Gangemi and Borgo 2004). As compared to covering the semantics of everything as in foundation ontologies, core ontologies provide a set of generic concepts whose semantics are shared across multiple domains (Deshayes, et al, 2007). Core ontologies not only provide the formal semantic of such concepts but they are also aimed at maximising the reuse and sharing abilities (Deshayes, et al, 2007). Therefore, core ontologies do not provide the semantic that are not shared by all the related domain ontologies (Deshayes, et al, 2007).

In 2004, a workshop was organised to find out the reasons for the success and failures of core ontologies. The workshop identified that there is need to work more on core ontologies to (1) reach an agreement on generic concepts in a community of practice (2) to dynamically communicate the semantics across a distributed

community (3) to align , map and merge various ontologies (4) to support the development of multiple applications or services (5) to define a generic template for specifying the content in some domain.

This thesis is directed to address the areas highlighted in point 2, 3 generally with specific focus being on points 1 and 4. One of the conclusions was that despite core ontologies not being widely used, there is a definite need to work more towards developing core ontologies (Gangemi and Borgo 2004).

2.4.2.2.3 Domain Ontologies

Ontologies developed for a certain domain with all the concepts in it dependent on the respective domain are called domain ontologies (Borgo and Leitão 2007). For example, product design could have its own domain ontology and production could have its own domain ontology.

Heavyweight domain ontologies capture the semantics for a particular domain. Multiple domain ontologies developed on their own do not share any common basis and therefore interoperability is very limited across them. However, the whole purpose of interoperability is to communicate and share information and knowledge across multiple domains. This highlights the need to develop core ontologies that can be used as a common basis for interoperating domain ontologies.

2.4.3 Ontology Development Methodologies

An ontology development methodology consists of the steps involved in the development of an ontology. Ontologists have been developing and using different methodologies for developing ontologies in accordance with the requirements. This section presents a review of the different ontology development methodologies to help define a methodology for developing the ontology being explored in this thesis.

2.4.3.1 IDEF5 Ontology Development Methodology

The IDEF5 ontology construction methodology involves the following five steps (IDEF5 Method Report, 1994).

1. Organizing and Scoping: The purpose, viewpoint and context of ontology are established and roles are assigned to the team members.
2. Data Collection: The raw data needed for the development of the ontology is collected from the domain or world of discourse.
3. Data Analysis: is done to extract the terms, concepts and relations for ontology. Existing ontologies should also be analysed for borrowing the useful concepts and relations rather than re inventing the wheel.

4. Initial Ontology Development: A prototype ontology is developed from the analysed data.
5. Ontology Refinement and Validation: Involves the refinement and validation of the raw ontology again and again until a satisfactory ontology conforming to the requirements is achieved. This step adds standardization and mapping base to ontology and makes sure that it is compliant with the standards and can be mapped and that there are no conceptual mismatches.

2.4.3.2 Blomqvist and Ohgren's Methodology

Blomqvist, Ohgren, (2008) presented a review of different ontology development methodologies and proposed their own method based on the study. Blomqvist and Ohgren's (2008) methodology is summarised in Figure 2.7. It consists of three main steps, (1) Requirement Analysis (2) Ontology construction (3) Evaluation and testing. The requirement analysis is carried out to clearly define the requirement of the ontology. It should outline the users and the uses, the purpose and scope, the specific task (functionality of ontology) and resources (ontology language and tools, knowledge sources and personnel). The requirement analysis should clearly document what is going to be in the ontology and what is not. The requirement document is reviewed to reach a consensus on the meaning of the terms to avoid conflicts. The collection of all this information is combined into a finalised requirements document (Blomqvist and Ohgren, 2008).

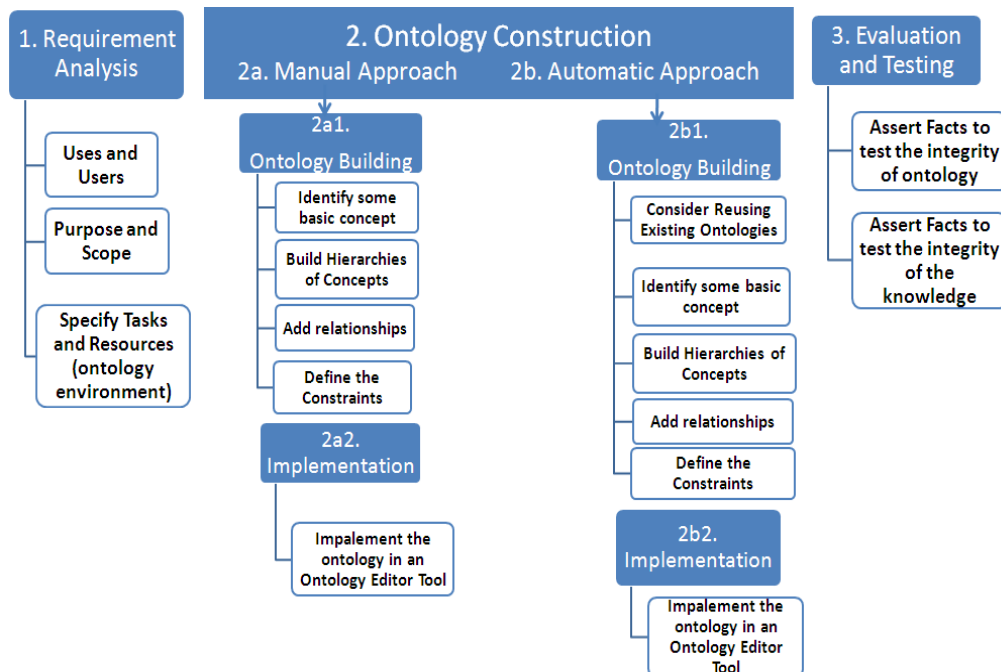


Figure 2.7: Blomqvist and Ohgren's Ontology Development Methodology (2008)

The 2nd and perhaps the most important step in Blomqvist and Ohgren's (2008) method is of ontology construction. This is further sub divided into manual and automatic approaches. Each approach has two steps i.e. ontology building and implementation.

Both approaches have the same steps involved other than the step involving the reuse of existing ontologies which is only present in automatic approach. The difference between the two approaches is that the automatic approach exploits the use of software tools to automatically identify the basic ontology structure from the requirements documents and existing ontologies.

In automatic approach, more than one ontology structures may also be identified which can be included in the ontology as a whole or by parts. Terms and relations for ontology are partly extracted from the requirement document with or without the help of software tools e.g. Text-To-Onto (Maedche 2003). The resulted terms are matched against the pattern/s. Software tools can be used to do it automatically (Cohen et al. 2003).

The manual approach on the other hand is based on the manual identification of concepts, relations and thus the structure of the ontology. After the basic hierarchies are built, the constraints on the meanings of concepts and relations are identified. The terms, relations, and constraints are collected in another document separately to avoid the problems relating to the language expressivity (Blomqvist and Ohgren, 2008).

The final ontology is build on the selected pattern which best suits the requirements and gives more comprehensive details and structure. More terms, relations, constraints and axioms may be added to improve the ontology further. The developed ontology can be tested on the systems like KAON (2005) or Siemens test bed system.

Automatically constructed ontologies give lesser details then the manual ones but these are more concrete and compatible (Blomqvist and Öhgren 2008). A combination of both was recommended by Blomqvist and Ohgren as the most suitable method for building ontologies. However, each approach may find its suitability more appropriate in the context of different works. The manual approach is suitable for developing the core ontologies and defining the constraints on concepts explicitly. Therefore, for the ontology being explored in the research work presented in this thesis, manual approach is preferred.

2.4.3.3 Noy and McGuinness Methodology

Noy and McGuinness' ontology development guide consists of the following seven steps(Noy and McGuinness 2000).

Step 1. Determine the domain and scope of the ontology

What is the domain that the ontology will cover?

What the ontology is going to be used for ?

What types of questions the information in the ontology should provide answers for?

Who will use and maintain the ontology?

Step 2. Considering the reuse of already present ontologies

Step 3. Enumerating key terms

Step 4. Defining the classes and class hierarchies

This can be done by following a top-down or bottom up approach or even a combination of these two.

Step 5. Defining the properties/attributes of classes

"Intrinsic" properties representing internal attributes of classes, "extrinsic" properties representing external attributes of classes, "parts" both physical as well as abstract and relations

Step 6. Define the facets of the slots

Slot cardinality

Slot-value type e.g. String, Number, Boolean, Enumerated, Instance

And Domain and range of a slot

Step 7. Create instances

Noy and McGuinness's methodology is quite comprehensive and in general suits many ontology development requirements. However, an additional step involving the definition of constraints on the concepts is required in developing heavyweight ontologies.

2.4.3.4 METHONTOLOGY

The methodology for the conceptualization of domain ontologies by the name of METHONTOLOGY was presented in 1997(Fernandez-Lopez et al. 1997). It was modified and improved later 1999(Fernández-López et al. 1999). METHONTOLOGY was developed for the software life cycle processes. It facilitated the project management process by providing the methodology for activities like planning, control, quality assurance of projects. It also facilitated the development of ontologies by guiding through conceptualization of domain, formalising the ontology, using the ontology and, guided in implementing the ontology. Moreover, it supported the

ontology evaluation integration with other ontologies and knowledge bases and assisted in knowledge acquisition, evaluation and documentation etc.

2.4.3.5 CommonKADS Methodology

The CommonKADS methodology (Schreiber et al. 2000) can be used to derive basic ontology building principles. It defines the basic principles for the development of knowledge based systems. This methodology resulted in a series of international research and application projects.

A couple of models have to be created to structure the Knowledge systems. Further three models are proposed to be developed at the “context” level of abstraction which are (1) Organizational model, (2) Task model and (3) Agent’s model. The organizational model is concerned with the problems, conflicts, opportunities and improvement areas of knowledge management. The task model is a description of all the tasks, activities or processes that can be performed by the organization. Tasks are executed by the agents so the Agent’s Model describes the entities, object and things which perform or can perform the task, activity or process it also describes their competencies and limitation.

The contextual level lies beneath the conceptual level and the conceptual level deals with the Communication model and the Knowledge model. These models are created and derived from the Organizational model, Task model and Agent’s model described above. The knowledge model deals with the required knowledge for performing a task. The communication between various agents performing the tasks is handled through the communication model. And last but not least, the structure of the knowledge system under the process of creation is described through the Design model.

2.4.3.6 Review of Other Ontology Development Methodologies

Various different approaches are found to be used by the researchers in the literature for developing ontologies and these approaches have evolved gradually with time. Lenat and Guha described the general steps and useful and interesting points regarding the development of the CYC ontology (Lenat and Guha 1990).

The first guidelines for developing an ontology were proposed by Uschold and King (1995) and Gruninger and Fox (1995) out of their experience of developing the Enterprise Ontology and the TOVE (TOronto Virtual Enterprise) project ontology. The guidelines were later refined, revised and updated (Uschold and Jasper 1999; Uschold and Gruninger 1996).

An ontology development methodology was proposed and used in ESPRIT KACTUS project (Schreiber et al. 1995). Swartout, et al (1997) proposed a novel method for building ontologies based on the SENSUS ontology. A few years later, Staab, et al (2001) presented the On-To-Knowledge methodology from the On-TO-Knowledge project. All of these methods & methodologies do not propose or take into consideration the construction of ontologies through collaboration and distribution (Corcho et al. 2003).

The method presented by Euzenat contained a proposal for the construction of ontologies through collaboration. This method presented a protocol for the agreement of new knowledge with already present knowledge and the knowledge architecture which has the history of agreement (Euzenat 1996). A detailed comparative study of these methods is given by Fernandez-Lopez (1999). The methods and methodologies being introduced so far only deal with the building of ontologies.

There are however, many other methodologies which can handle different areas like ontology reengineering handling methodology (Gomez-Perez and Rojas 1999), Ontology learning (Aussenac-Gilles, 2000 and Kietz, et al 2000), Ontology evaluation e.g. the knowledge verification framework (Gomez-Perez 1996 and 2001) and Ontology analysis (Guarino and Welty's 2000), A formal ontology of properties by Guarino and Weltys (2000), Managing ontological constraints by Kalfoglou & Robertson (1999), Use of formal ontologies to support error checking in specification by Kalfoglou & Robertson (1999).

2.4.3.7 Mapping and Merging of Ontologies

Ontology mapping and verification is crucial for cross domain interoperability to make sure that what is being interpreted by one partner is actually what was meant by the other (Anjum, 2011). This in turn facilitates correct and effective knowledge sharing across multiple domains.

There are three ways of making the heterogeneous ontologies interoperable are identified as: (1). by building the inclusion relations between different ontologies, (2). by building the mapping relations between different ontologies and (3). by building a standard or common ontology from the various local ontologies (Liping, et al, 2007).

In order to get rid of the lengthy and tiring work to align, merge or map various ontologies together for the purpose of Interoperability, it is desired to have some tools or agents that can support these activities. There are some software tools which can be used for semi automatic mapping and merging of ontologies (Noy 2004).

The ontology mapping and verification is a fully exploitable research area, however, the research work reported in this thesis is focused on the development of a core ontology to support knowledge sharing across product design and production domains. It is anticipated that this will lead to effective mapping and verification methods which have been the subject of other study in the IMKS project (Anjum, 2011 Anjum et al 2010).

2.4.4 Ontology Development Languages and Tools

Ontology development languages are used for representing the structures that constitute ontologies and provide the bases for capturing and reasoning about knowledge. Examples of these are UML, OWL and KFL. Ontology development tools refer to the software applications that facilitate the use of ontology development languages for building ontologies e.g. IODE, Protege and Enterprise Architect.

2.4.4.1 Ontology Development Languages

Ontology development languages can be broadly classified into three main categories: (1) ontology mark-up languages (2) schematic languages and (3) general ontology languages.

Figure 2.8 illustrates these categories with the help of examples. Ontology mark-up languages like the Resource Description Framework (RDF), RDF Schema (RDFS)

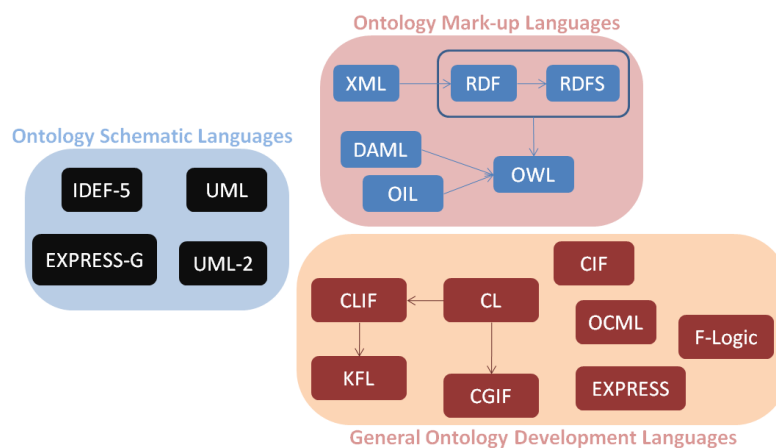


Figure 2.8: Categories of ontology development languages

and the Web Ontology Language (OWL) have their syntax based on the eXtensible Mark-up Language (XML) and are Description Logic-based. Description Logics (DL) is a subset of First Order Logic (FOL) which is optimised to guarantee decision making for inference engines. Schematic languages provide a way of representing the ontologies in a graphical or diagrammatic way. General ontology languages, on other hand, are largely based on FOL and offer the potential for developing

heavyweight ontologies. Examples of these languages are Knowledge Interchange Format (KIF), Common Logic (CL) (ISO 24707, 2007) and KFL (Highfleet, 2012).

Table 2.3 presents a summary of different ontology development languages in terms of their descriptions, the level of formalisation they offer (i.e. lightweight and heavyweight), and the developers who were the main contributors in the development of those languages. A detailed review of the ontology development languages relevant for the research work reported in this thesis is presented next.

Table 2.3: Different ontology development languages

Ontology Development Language	Description	Formalisation	Developer
XML	eXtensible Markup Language provides the syntax for the stack of Semantic Web languages	Lightweight	World Wide Web Consortium (W3C)
RDF, RDF Schema, RDF(S)	Resource Description Framework is used for modelling information to be deployed as Web resources	Lightweight	World Wide Web Consortium (W3C)
OWL	Web Ontology Language is a markup language to develop ontologies for the Semantic Web	Heavyweight	World Wide Web Consortium (W3C)
IDEF5 Schematic Language	Integration DEFINition Schematic Language is used to design ontologies	Lightweight	Knowledge Based Systems, Inc.
UML and UML-2	UML is an object-oriented graphical notation to model information. UML-2 can be exploited for representation of ternary and higher order relations and can, therefore, be used for the lightweight view of CL based ontologies as well (Claire et al, 2012). Java and C, C++ etc codes can be generated from UML2 diagrams that cannot be done with IDEF-5.	Lightweight	Object Management Group
EXPRESS-G	Diagrammatic notation for information models	Lightweight	ISO 10303
EXPRESS	A language for modelling product data	Lightweight	ISO 10303
F-Logic	Frame Logic is an object-oriented approach to FOL used for deductive and object-oriented databases	Heavyweight	Stony Brook and Mannheim Universities
OCML	Operational Conceptual Modelling Language is a language that supports the definition of 1st and 2nd order axioms	Heavyweight	Knowledge Media Institute, The Open University
KIF	Knowledge Interchange Format provides a syntax for FOL which can be understood by computers	Heavyweight	DARPA knowledge sharing effort
Common Logic (CL)	Common Logic is described as a framework for a family of languages which are logic-based	Heavyweight	ISO/IEC 24707
CGIF	Conceptual Graph Interchange Format is a dialect of CL used for representing conceptual graphs	Heavyweight	ISO/IEC 24707
CLIF	Common Logic Interchange Format is the KIF-like syntax used in CL	Heavyweight	ISO/IEC 24707
KFL	Knowledge Frame Language is based on an extended implementation of CLIF called ECLIF	Heavyweight	Highfleet, Inc.

2.4.4.1.1 Unified Modelling Language (UML)

The Unified Modelling Language (UML) provides a way for the modelling of knowledge and information. UML provides various diagrams like Class Diagrams, Use-Case Diagrams, and Communication diagrams. The most widely used ones are

the class diagrams. Figures 2.11, 2.12 and 2.15 presented later in the chapter are examples of UML class diagrams.

UML however had the issues of expressivity and platform independence as compared to IDEF-5 particularly in terms of representing ternary and higher order relations. Claire et al (2012) proposed that UML-2 can be exploited for the representation of ternary and higher order relations and can, therefore, be used for the lightweight view of CL based ontologies as well (Claire et al, 2012). Java, C and C++ etc codes can be generated from UML/UML-2 diagrams that cannot be done with IDEF-5. UML-2 is also the specified lightweight ontology development formalism for the IMKS project.

2.4.4.1.2 HTML & XML

The very early Mark up languages which laid the foundation for the development in the ontology of building languages is Hyper Text Markup Language (HTM). HTML is designed to represent data in an organised way using predefined tabs. The HTML tabs does not support logical reasoning which makes the extraction of data difficult (Gomez Perez, et al, 2008).

The eXtensible Markup Language (XML) addresses this problem by including the logical tabs in its syntax. In fact any tab can be added in XML to organise the data or information. The XML document might then be parsed into different formats such as the HTML for better appearance (Maruyama H., et al, 1999) and ease of understanding. Both the HTML and the XML do not have capability of capturing the formal semantics.

2.4.4.1.3 RDF & RDF(S)

The Resource Description Framework (RDF) and RDF Schema (RDFS) are methods for modelling data that offers the ability to develop hierarchies of objects and properties. It provides a framework for metadata description in the form of statements that are composed of object, attribute and value triplets. The object is called resource and can be represented by a Uniform Resource Identified (URI). The triplet can itself be either an object or a value. Value can represent either a string or resource and the relations between objects and values are represented by attributes. For example the sentence "*CuttingTool* has parameter *ToolDia*" can be represented in RDF as shown in figure 2.9.

RDF is not a language but a data representation model (Gomez-Perez, 2008). The RDF Schema (RDFS) provides a basic library, as set of classes and properties that provide the basic structure for developing the ontologies in RDF.

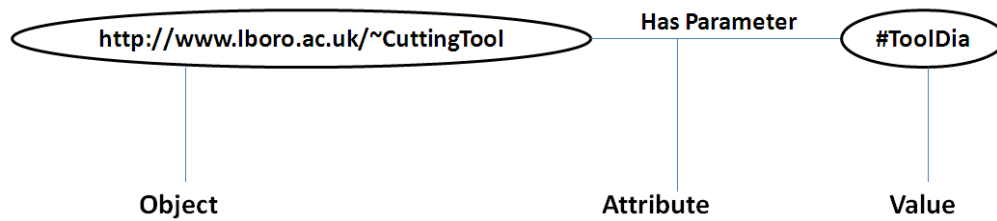


Figure 2.9: An object, attribute and value triplet of RDF

2.4.4.1.4 Web Ontology Language (OWL)

The World Wide Web Consortium (W3C) identified a need to define a language with more expressive power, ability to formalise the conceptualisations and reason over the information to support ontology building. Therefore, a new language was developed by combining the DAML and OIL languages into the Web Ontology language (OWL) (Gomez-Perez, 2008). OWL was designed to be used by applications that would not only present information to humans but would also process the formal content of information. OWL provided more machine interpretability than the XML, RDF, and RDFS. It does so through a richer vocabulary along with formal semantics (<http://www.w3.org/TR/owl-features>).

However, OWL is still limited in its expressive power. Because OWL is based on RDF/RDFS, it cannot relate more than two arguments in a statement. For example, using OWL the sentence “CuttingTool has parameter Tool Dia” can be modelled in a single statement as shown in figure 2.9. However, modelling a sentence like “Cutting tool has parameter Tool Dia that has a value of 30mm” will require at least three RDF statements with three arguments (Cutting tool, Tool Dia, and 30mm) and two relation/attributes (has Parameter and has Value). This is because RDF/RDFS/OWL cannot support the modelling of ternary or higher-arity relations. This translates into more extensive modelling, processing power and unnecessary complex structures.

2.4.4.1.5 Common Logic

Common Logic (CL) (ISO/IEC-24707 2007) is a formalism based on the first order logic that can formally represent the semantics of anything. Common logic has higher expressive power and supports better inference and reasoning ability as compared to the languages like XML, RDF/RDFS and OWL. It is also an international standard (ISO/IEC-24707, 2007) where a family of logic basic languages are provided as standard dialects of common logic. These dialects are languages recommended as a standard for formalisations in CL e.g. Knowledge Interchange Format (KIF), Common Logic Interchange Format (CLIF) and XML based Extensible Common Logic (XCL).

The formalism used to model ontologies in heavyweight logic in this thesis i.e. KFL is also based on common logic.

2.4.4.2 Ontology Development Tools

Corcho (2003) and Frankovic and Budinska (2006) presented detailed reviews of various ontology development tools. Based on the guidelines provided by these works, a review of ontology development tools relevant in the context of this thesis is presented in Table 2.4. The most relevant tool i.e. IODE is discussed in detail as this is the main ontology development environment for the ontology being explored in this thesis.

Table 2.4: Relevant ontology development tools

Ontology Development Tool	Description	Formalisation	Developer
Ontolingua	Supports ontology representation and sharing. It is based on KIF. Can translate to and from DL. Offers, browsing, creating, editing ontology. Does not have an inference engine.	Heavyweight	Knowledge Systems Lab, Stanford University
OntoEdit	OntoEdit is a graphical ontology editor and uses the KAON ontology management infrastructure (Gabel, 2004). It is based on the On-To knowledge methodology and CommonKADS methodology. It has inference and reasoning capabilities. It uses Ontobroker both as knowledge base.	Lightweight	http://www.semtalk.com (accessed 27/02/2012)
Protégé	Protégé is perhaps the most widely used ontology development tool. It supports ontology development languages like RDF and OWL. It has a customizable user interface. It also has powerful plug-in architecture which enables integration with other applications e.g. it can be integrated with the SWRL to write rules and axioms and thus make the ontology more rigorous. It has its limitations. Only up to binary relations can be defined in it. Offers low expressive power and inference and reasoning capabilities as compared to the CL based tools.	Lightweight	Knowledge Systems Lab, Stanford University
WebODE	This ontology engineering bench supports creation and editing of ontologies in an interactive and integrated way. WebODE is has a three tier architecture: (1) ontology environment (2) application server (3) database management level. (Julio et al. 2001). It has a client server architecture rather than a plug in which distinguishes it from Protégé and OntoEdit (Mizoguchi 2004). This architecture assists the utilization of existing services which provides high utilization of available resources (Mizoguchi 2004). WebODE facilitates import, export and translation between various ontology languages e.g. RDF(S), OIL, DAML+OIL, and Frame Logic (Mizoguchi 2004)	Lightweight	Ontology and knowledge reuse group, Technical University of Madrid
IODE	Integrated Ontology Development Environment (IODE 2010) is the only commercially available CL based ontology development tool. It uses an eXtensible Knowledge Server (XKS) as the database server which directly supports the ontologies build in IODE. IODE offers expressive power and inference capability that it inherits from being based on CL. It uses Simplified Common logic in the form of KFL. IODE is much more powerful than OWL or RDF/RDFS based tools having much more expressive and inference power. It has a built in mechanism for writing complex axioms. The ontology build in IODE can be directly loaded into the XKS. The XKS can then be populated with knowledge instances.	Heavyweight	Highfleet Inc.
MS Visio	MS Visio is multipurpose tool that supports UML, IDEF-5 and other diagrammatic representations. This tool is totally informal and doesn't capture any understanding of semantics. This is, therefore, only used for developing diagrams of ontological concepts and relations.	Lightweight	Microsoft
Enterprise Architect	Enterprise Architect is software tool for developing the lightweight UML representations of the ontologies. Enterprise Architect allows the representation of ternary and higher order relations which makes it suitable for modelling ontologies involving ternary and higher order relations. It also allows the modelling of power types and clabjects (discussed in chapter 7) which distinguishes it from other UML software tools like Altova.	Lightweight	Sparx Systems Pty Ltd

2.4.4.2.1 IODE & XKS

The Integrated Ontology Development Environment (IODE 2010) is the only commercially available Common Logic based ontology development environment. The eXtensible Knowledge Server (XKS) which is also developed by Highfleet Inc., directly supports the ontologies built in IODE. It provides knowledge base for holding

and populating the facts. The IODE uses the Knowledge Frame Language (KFL) which is based on the Extended Common Logic Interchange Format (ECLIF). This provides the ontologies built in IODE better expressive and inference capabilities than OWL based tools. The ontologies built in IODE can be directly loaded into the Knowledge Base in XKS. The XKS can then be populated with knowledge instances using the simplified Common Logic.

The IODE uses a foundation ontology called Upper Level Ontology (ULO). The ULO provides predefined generic concepts and relations like *Duration*, *Concrete entity*, *Abstract Entity*, *Real Number*, *Binary Relation* and *Transitive Binary Relation*. The ULO provides the generic conceptualisation for the Entities, Processes and Relations.

2.4.5 Purpose and Uses of Ontology

The applications of ontologies range across various fields from medical sciences, web designing, processing industries, construction industry and manufacturing. The purpose of ontologies in general is to provide a common basis or shared understanding. Ontologies may be used in the computing world for the purposes of communication, computational inferences, information & knowledge organization, exchange and reuse (Gruninger and Lee 2002).

An ontology, as common basis for shared meanings, can be used for the purpose of knowledge sharing and interoperability across multiple domains. Three main categories of uses for ontologies in concurrent engineering including enterprise modelling and multi-agent systems are 1. Communication (between different systems and system components), 2. Interoperability (between systems using ontologies) and 3. systems engineering (providing support to the design and software development) (Roche 2000).

Ontologies play a pivotal role in knowledge management by providing a better way of representing knowledge and supporting the development of reusable and shareable knowledge bases (Sureephong et al. 2008). One of the main purposes of ontology is to explicitly define the entities, their attributes and relations to produce an interoperable knowledge format that is understood by both humans and machines, thereby achieving the goals of knowledge sharing, reuse and interoperability (Lin and Harding 2007; Uschold and Gruninger 1996).

2.5 Ontologies in Manufacturing

Ontologies have been used in various manufacturing systems for developing the structures and hierarchies of concepts and defining the relations between various

concepts and systems. Examples of manufacturing ontologies (or systems that have ontological structures) are; Model Oriented Simultaneous Engineering System (MOSES) (Molina and Bell R 1999), CIM Open System Architecture (CIMOSA) (CIMOSA-Association 1996; ESPRIT-Consortium-AMICE 1993; Kosanke et al. 1999), MISSION's intelligent manufacturing system project (MISSION-Consortium 2001; Rabe 2000), Product Configuration Model by Yang, et al (2009), STEP Standard (ISO-10303 2006), PSL (ISO 18629 2004), ADACOR (Borgo and Leitão 2007), SMIF and (Chungoora, 2010).

A comprehensive review of the ontologies is given in the following sub sections. Table 2.5 presents a list of ontologies shortlisted from the reviewed ontologies. The

Table 2.5: List of ontologies shortlisted from the reviewed ontologies

Sr. No.	Authors	Year	Ontology	Relevance to the MCCO	LW / HW
1	Gunendran and Young	2010	Capturing best practice manufacturing Knowledge	Very High	LW
2	Chang et al	2010	Design for Manufacture (DFM) ontology	High	HW
3	Chungoora	2010	Semantic Interoperability Framework	Very High	HW
4	Chen et al	2009	Development of a mechanism for ontology-based product lifecycle knowledge integration	Medium	LW
5	Tursi, et al	2009	Product Ontology for embedding product information for interoperability	Medium to High	LW
6	Dartigues et al	2008	Ontology of Features to support CAD/CAPP Integration	Very High	HW
7	Blomqvist and Ohgren	2008	Ontology for Model-based Software Engineering of Dependable Systems (SEMCO) project	Medium	LW
8	Chang, et al	2008	An ontology-based support for product conceptual design	High	LW
9	Lee and Suh	2008	ontology-based multi layer knowledge framework (OMKF)	Very High	HW
10	Lagos & Setchi	2007	Manufacturing Ontology for e-learning	Low	LW
11	Abdul-Ghafoor, et al	2007	A Common Design Feature Ontology for Product Data Semantic Interoperability	High	HW
12	Terzi, et al	2007	Holonic metamodel for product information traceability	High	LW
13	Semre, et al	2007	Machining Ontology	very High	HW
14	Borgo and Leitao	2007	ADaptive Holonic COntrol aRchitecture for distributed manufacturing systems is the architecture (ADACOR), Core	Very High	HW
15	Fenves, et al	2006	Core Product Model (CPM)	High	LW
16	Leimagnan, et al	2006	MANufacturing Semantics ONtology (MASON)	High	HW
17	Sarder	2006	Design ontology	Medium to High	HW
18	Patil,et al	2005	Product semantic representation	Medium	HW
19	Harding et al	2004	Manufacturing Systems Engineering (MSE) ontology	Medium	LW
20	Martin and D'Acunto	2003	the product process integration model	Very High	
21	Schlenoff et al	2000	Process Specification Language (PSL)-ISO-18629	Medium	HW
22	Kosanke et al	1999	Computer Integrated Manufacturing Open System Architectur (CIMOSA) ontology	Medium	LW
23	Hardiing and Yu	1999	Factory Design Model (FDM)	Medium	LW
24	Molina and Bell	1999	Manufacturing Model for (MOSES) project	High	LW
25	Uschold et al	1998	Enterprise Project	Medium	LW
26	Fox & Gruninger	1997	TOVE project	Medium	HW
27	ISO Standards		ISO-10303-1,49, 224, and 239, ISO-13584, ISO-15531-1 (MANDATE), ISO19439, ISO/IEC-TR-10000-1, ISO/TS-10303-1017, 1018, 1022, 1164, ISO-13399, ISO-15531-43,	Medium to Very High	LW

reviewed ontologies are mostly from within the area of manufacturing. However, ontologies from other domains that are useful for the present research in terms of the idea development, research approach and the content of ontology have also been studied. The discussion that follows focuses on the ontological research most relevant to this research.

2.5.1 Manufacturing Enterprise Level Ontologies

2.5.1.1 Lightweight Manufacturing Enterprise Level Ontologies

This section presents a review of the lightweight manufacturing ontologies as well as the models and standards that hold lightweight ontological structures at manufacturing enterprise level.

2.5.1.1.1 CIM Open System Architecture (CIMOSA)

CIM Open System Architecture (CIMOSA) (CIMOSA-Association 1996; ESPRIT-Consortium-AMICE 1993; Kosanke et al. 1999) was targeted at the enterprise modelling and inter enterprise interoperability between manufacturing enterprises. The structure provided as part of the CIMOSA can be reviewed as a lightweight ontology. It aimed to provide a common understanding for making transparent the enterprise knowledge and business processes and assisted in the provisions of a better decision support system. It highlighted the need to make enterprise models to be made useable by operational staff rather than external consultants to keep the models up to date and more useful. The ontological structure of CIMOSA supported the capture of knowledge about *Enterprises*, *Virtual Enterprises*, *Supplier*, *Customer*, *Orders*, *Products* and *Parts* and their relations to assist interoperability. The CIMOSA, therefore, can be useful in developing an understanding of the concepts like *Manufacturing Enterprise*, *Products*, *Parts* and can also assist in defining their relations.

2.5.1.1.2 Factory Design Model (FDM)

Harding and Yu (1999) developed a Factory Design Model (FDM) which could provide information regarding the competence of a proposed factory and retrieve the information regarding an already present factory. It was shown that by using these two critical set of information the factory design process can be enhanced, errors can be reduced, and time and money can be saved (Harding and Yu 1999).

However, like CIMOSA, FDM ontology was also not focussed effectively on product design and production domain and remained at the manufacturing factory's design level. FDM also has an underlying structure based on lightweight ontologies which has some useful manufacturing concepts like *Manufacture*, *Production*, *Production*

Plan, *Material* and *Product*, but these were identified and defined in a factory design context.

2.5.1.1.3 The Use of Standards as Ontologies

Several ISO standards mainly from the technical committee 184 and sub-committee 4 are reviewed. Most of these standards can be considered as lightweight or non formal ontologies. These standards provide a number of useful concepts and their textual definitions can be useful for developing a core heavyweight manufacturing ontology (Young et al, 2007). Examples of such concepts are *ManufacturingFeature*, *Part*, *Resource*, *Process*, *Component*, *ManufacturingMethod*, *ProcessPlan* etc. Other such concepts from standards are discussed in detail in chapter 5, 6, and 7. However, the concepts in standards have their issue when it comes to formal capture of semantics.

One issue with these standards is the text based and non formal definitions of concepts. The textual definitions mean that the system cannot interpret the semantics of concepts. This means that standards are very useful individually within their narrow domains of application where meanings are understood by the concerned community. However, for cross domain interoperability, standards can lack semantic integrity and consistency.

Another issue in standards is regarding the variations in the definitions of the same terms. The definitions of terms in ISO standards vary not only across different standards but within different modules of the same standards (Usman et al. 2011).

2.5.1.1.4 Ontological Integration of Product Lifecycle Knowledge

Chen et al (2009) proposed a novel mechanism for integration of ontological product lifecycle knowledge from the different enterprises. Chen et al (2009) proposed an approach where different enterprise ontologies were combined and then decomposed into different ontologies representing different categories of concepts within the product lifecycle. Then those ontologies were merged into a single global ontology which could support knowledge sharing across multiple domains for all the concerned enterprises. In order to support the development of global ontology a generic global ontology was defined which could be extended and adapted to support integration of different domain ontologies. The approach is partially based on the dynamic approach to interoperability because the global ontology adapts dynamically to support interoperability across new domain ontologies.

Chen et al (2009) work enhanced the understanding of the ontological approaches for knowledge sharing. The approach and the proposed ontology was tested for knowledge sharing between Mould design and Mould production domains.

The generic global ontology provides help in categorising the concepts for product lifecycle and also provides some useful concepts like *Product Manufacturing*, *Part Description*, *Generic Process Planning*, and *Variant Process Planning*.

2.5.1.2 Heavyweight Manufacturing Enterprise Level Ontologies

The advantages of heavyweight ontologies over lightweight ontologies have partially been reported in section 2.4.2.1.2. Young et al, (2007) explained that there are advantages of using heavyweight ontologies for manufacturing knowledge sharing. In this section a review of the enterprise level heavyweight ontologies is presented.

2.5.1.2.1 Enterprise Project and TOVE Project

Research projects like the Enterprise Project (Uschold et al. 1998) and the TOVE project (Gruninger and Fox 1995) aimed to develop a Common Sense Enterprise Model which could support reasoning at shallow level knowledge between manufacturing enterprises as well as virtual enterprise. In order to do that, the ontological approach was used for the development of hierarchies and taxonomies. It is understood that these projects were too generic for effective use in product design and production domains. However, the method of defining the competency questions and verifying the developed ontologies against them can be useful in defining the logical constraints for the concepts in the ontology being explored in this thesis. In particular the axioms for planning and scheduling can be helpful in developing a generic understanding of process planning concepts.

2.5.1.2.2 Manufacturing Systems Engineering (MSE) Ontology

MSE moderator developed by (Harding, J.A., et al, 2003) as part of the MISSION project (MISSION-Consortium 2001) also has an underlying structure based on a lightweight ontology. The ontological concepts and relations are more focussed on the manufacturing systems at the enterprise level. This work was extended later by Lin and Harding (2007) into a MSE model containing a high level manufacturing ontology. The MSE ontology served as a common meta model to support interoperability across extended project teams at inter enterprise level (Lin and Harding 2007) while allowing the partners to keep their preferred models.

The MSE ontology and its approach is illustrated in figure 2.10. The MSE ontology defined useful and shared concepts like *Resource*, *Production Resource*, *Production*

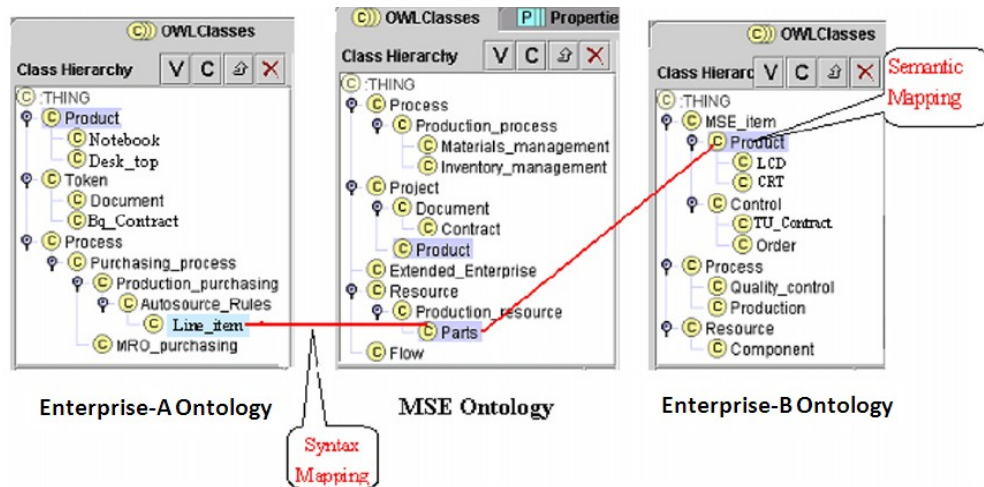


Figure 2.10: MSE ontology as a mediator between different enterprise ontologies

Process, *Product*, *Process* and *Parts* to act as a mediating ontology for interoperability across different enterprise ontologies. Such concepts as well as the approach of defining the MSE ontology as a mediating ontology for supporting interoperability across different enterprise ontologies can be useful for defining the core concepts being explored in this thesis.

The MSE ontology was, however, too generic to support interoperability across design and production because of being oriented towards inter enterprise interoperability. Moreover, the semantics of concepts in the MSE ontology were not defined rigorously enough to support interoperability through the system. This meant that the mapping of different MSE ontologies to the MSE ontology had to be done manually which consequently translates into large overhead. Lin and Harding (2007) suggested that such overheads can potentially be reduced by automatic mapping tools as well as through better captured formal semantics.

2.5.1.2.3 Process Specification Language

Process Specification Language (PSL) (Schlenoff et al. 2000) developed by NIST is perhaps one of the best available heavyweight processes ontologies. PSL was initially developed to capture the semantics of manufacturing processes but its generic nature made it suitable to capture the semantics of most processes. PSL was also established as a standard (ISO-18629-1 2004) for capturing the process semantics. PSL provides a high level language to support semantics interoperability across different process domains. The PSL concepts like *Activity*, *Activity Occurrence*, *minPrecedes* and the formal theories for these concepts can be really

useful in formally defining the process semantic for different concepts in the ontology being explored in this research.

PSL, however, is limited in its ability to define the objects and auxiliary concepts needed for finer details (Niles 2001; Schlenoff et al. 2000) and is also limited in linking the resources and tangible inputs and outputs to the processes (Young et al. 2007).

2.5.1.2.4 Semantic Interoperability between Application Ontologies

Patil et al (2005) presented an ontological method for the formal capture of product semantics and supporting interoperability of product information. An ontology namely Product Semantic Representation Language (PSRL) was built using the formal languages of DAML+OIL. Based on the logic embedded in the PSRL, different application domain ontologies were aligned with the PSRL. Once the ontologies are aligned with the PSRL, interoperability across them is possible (Patil et al. 2005a; Patil et al. 2005b). The common semantics of PSRL ontology provided a base for establishing semantic mapping between the application ontologies. The PSRL ontology was modelled using core concepts and relations from feature based modelling systems. Therefore, the PSRL could only be used for interoperability of product semantics between different CAD systems e.g. between 'Unigraphics' and 'SolidWorks' CAD systems.

Only the concept of *Feature* can be of use for the explored ontology. Other concepts like *BaseExtrudedSolid* and *BaseRevolvedSolid* can be useful for interoperability across CAD systems. The PSRL was unable to capture and exchange important product information relating geometric features like points and lines (Patil et al 2005). Patil et al (2005) recommended the use of more expressive common logic based languages like KIF for complete formal representation and exchange of product data. Therefore, the work was useful in developing an understanding of the use of common logic based languages for enhanced formal expressiveness.

2.5.2 Detailed Manufacturing Ontologies

2.5.2.1 Lightweight Detailed Manufacturing Ontologies

This section provides a review of ontologies that use a lightweight ontological approach and that are either narrowly focused on the manufacturing or can directly support the development of a core manufacturing ontology.

2.5.2.1.1 Model Oriented Simultaneous Engineering System (MOSES)

Molina and Bell (1999) presented a manufacturing model as part of a project called Model Oriented Simultaneous Engineering (MOSES). The structure of the MOSES

model is based on a lightweight manufacturing ontology. The work was designed to capture the manufacturing capability of manufacturing industry. The model consisted of three very important categories of concepts i.e. *Manufacturing Resources*, *Manufacturing Processes*, and *Manufacturing Strategies* (Molina and Bell R 1999). The manufacturing capability based on the above constituents was captured at four levels i.e. *Factory*, *Shop*, *Cell* and *Work Station* levels. The work is one of the key ones in the journey towards a heavyweight manufacturing ontology by contributing towards an improved understanding of useful concepts like *Factory*, *Shop*, *Cell*, *WorkStation*, *Manufacturing Resource* and *Manufacturing Process*. The limitation of the works is that it only captured the manufacturing information and not aimed at sharing it across to other product lifecycle domains. The lightweight nature of the work does provide some key concepts but with no formalisation the ability of system to interpret these terms is very limited.

2.5.2.1.2 Holonic Approach to Manage Product Lifecycle Data

Holonic approaches in manufacturing are based on the famous work of Koestler (1967) where 'holons' are used to model behaviours of entities whereby they can behave as a part as well as a whole. Terzi et al (2007) presented a holonic approach for the traceability and management of product lifecycle data. A holonic Meta model (which can be considered to be a lightweight ontology) was developed to facilitate the information traceability of product, along its lifecycle (Terzi et al. 2007). The holonic product model was developed in UML and XML to define the hierarchies of concepts and their relations. Using the holonic product model, information can be traced back and forth from holonic machines, tools and models etc. This can reduce production time and therefore cost. The proposed Meta model has partly reduced the problems of information exchange using holonic concepts.

Terzi et al (2007) work provides an understanding of how the generic concepts like *Product Development*, *Product Production*, *Product Use*, and *Product Dismiss* provide a basic structure for tracing (exchanging) the detailed information from product lifecycle information.

2.5.2.1.3 A Product Ontology for Integrating Production Planning and Design

Tursi et al (2009) proposed a novel method of exchanging the product information by treating the product itself as an interoperable system (Tursi et al. 2009). A product ontology was proposed as a common model to integrate production planning systems with product design applications. Tursi et al's (2009) work was oriented specifically at the transformation of Bill of Materials (BOMs) with particular focus on transforming

engineering BOMs into manufacturing BOMs. This was achieved by embedding the product ontology into the product models. The product ontology was built by borrowing concepts from the standards relating to the product data management (ISO/TS-10303 2004) and enterprise integration (ISO/IEC-6224 2002). The product ontology that was embedded into the product models helped to minimise the issue of semantic ambiguities and mismatches (Tursi et al. 2009). Domain rules were defined to enable the mapping of engineering BOMs with the product ontology and the mapping of product ontology with manufacturing BOMs.

Tursi et al's (2009) work, being focused on BOMs, was oriented towards assembly areas of production planning domain and design domain. However, the inclusions of concepts concept like *Part*, *Product*, *Component*, and *PartVersion* from the ISO standards into Tursi's product ontology can be useful in identifying concepts from standards that can be incorporated into the manufacturing ontology being researched.

2.5.2.1.4 A Model to Share Manufacturing Best Practice Knowledge

Gunendran and Young, (2010) presented a method to capture best practice manufacturing knowledge. They used best practice libraries, product models and the manufacturing models to support this. This work showed the effectiveness of features and part families in organising and capturing manufacturing knowledge. This also explored the relations between design and manufacturing features as well as part families.

They highlighted the issues of scaling of their approach to an industrial scale. Issues of knowledge maintenance and development of future PLM tools were also highlighted. One of the important issues highlighted was regarding the exploitation of this approach to identify the production consequences in the product design phase. They also highlighted the need to explore further the relations between design and manufacturing.

2.5.2.1.5 The Core Product Model

Core Product Model by NIST (Fenves et al. 2006) provided a lightweight ontology to capture the product model data. It has some useful core manufacturing concepts particularly the feature concepts. The work is isolated from many existing manufacturing models and needs to reconsider the use of existing ontologies for their exploitation in developing a new and more comprehensive product model. The work only focuses on capturing product model and the ability to share knowledge across domains has not been represented. However, the CPM can be very useful in defining the core manufacturing concepts such as *Feature*, *Form* and *FormFeature*.

2.5.2.1.6 An Ontology Based Tool for Product Data Exchange

Chang, et al, (2008), presented an ontology based tool to help design products and share and exchange useful design knowledge with the designer to save time and money. The ontology defined in this work was defined using the IDEF methodology which meant that the ontology was lightweight in nature. The research contribution by Chang et al (2010) can be helpful in developing an understanding of design concepts like *Function* and its relation to the concepts like *Component* and *Products*. Where *Function* and *Component* contribute to the development of a *Product*.

Chang et al (2008) highlighted the need to work more towards ontologies for exchanging product information to overcome the limitations of the present database systems in capturing complex relations and support interoperability across different domains. Chang et al (2008) also recommended the extension of their work to enable exchange of information from manufacturing with product design.

2.5.2.2 Heavyweight Detailed Manufacturing Ontologies

2.5.2.2.1 Manufacturing Semantics Ontology (MASON)

The Manufacturing Semantics Ontology 'MASON' (Leimagnan et al. 2006) was developed to enable the formal capture of semantics of concepts related to the manufacturing industries. A lightweight representation of the main set of classes in MASON is shown in figure 2.11. The semantics of these classes and their relations were captured in formal logic using OWL. Use of OWL makes MASON more widely applicable and interoperable. Using MASON, the production knowledge related to individual operations can be formally captured.

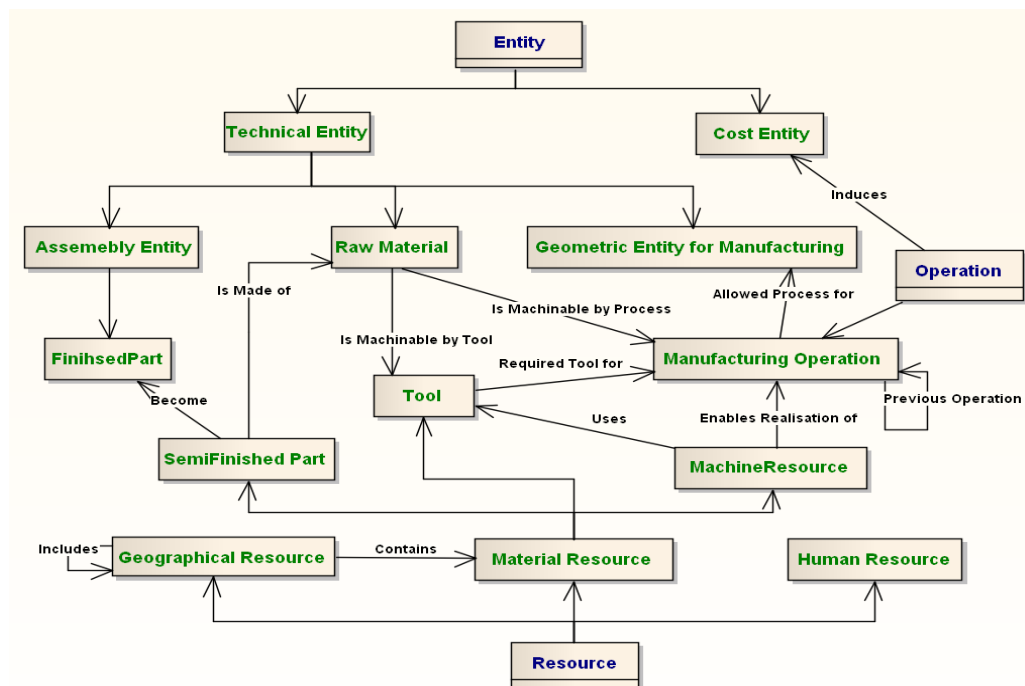


Figure 2.11: Main concepts of MASON reproduced from (Leimagnan et al. 2006)

The ontology and proposed approach in MASON, however, does not provide a method for sharing the production knowledge with product design. The ontology lacks the feature and part family perspective that makes it difficult to capture and relate the knowledge to a particular feature or part. The concepts and relations in MASON (as shown in figure 2.11) can potentially facilitate the development of the understanding of the concepts and relations for the researched manufacturing ontology.

2.5.2.2.2 A Machining Ontology Based on MASON

A machining ontology developed by Semere (2007), exploited MASON and extended it to define the concepts and relations for the machining domain (Semere et al. 2007). As a result of being based on MASON, the ontology by Semere et al (2007) addressed the issues in formal capture of semantics of concepts. The ontology also considered a feature aspect for defining the concepts for the machining domain.

The part family aspect is however missing in Semere's ontology which means that the machining knowledge about features is captured independently of the part or part family to which they may belong. A drawback of the ontology is that the context of sharing production knowledge into product design has not been considered in this ontology. This ontology does support the capture of machining knowledge but the ability to share this into product design or other product lifecycle domains remains to be explored.

2.5.2.2.3 ADACOR ontology

Borgo and Leitão (2007) developed an architecture i.e. ADaptive Holonic Control architecture (ADACOR) to support the control of manufacturing planning and scheduling activities for distributed manufacturing systems. As part of the ADACOR architecture, a core manufacturing ontology containing core manufacturing planning and scheduling concepts was defined. The ADACOR manufacturing ontology was well founded because it adopted the formal semantic base from DOLCE foundation ontology and captured the semantics of core production planning and scheduling concepts in formal logic. The set of concepts in the ADACOR ontology, as shown in figure 2.12, and the formal definitions of those concepts may prove to be useful for the ontology being researched. Also, Borgo and Leitão's (2007) work can be useful in developing an understanding of the use of core ontologies to support knowledge capture, retrieval and sharing in general.

ADACOR ontology is mainly focused on manufacturing control and scheduling activities, therefore, the core manufacturing ontologies needs further exploration to support knowledge sharing between product design and production.

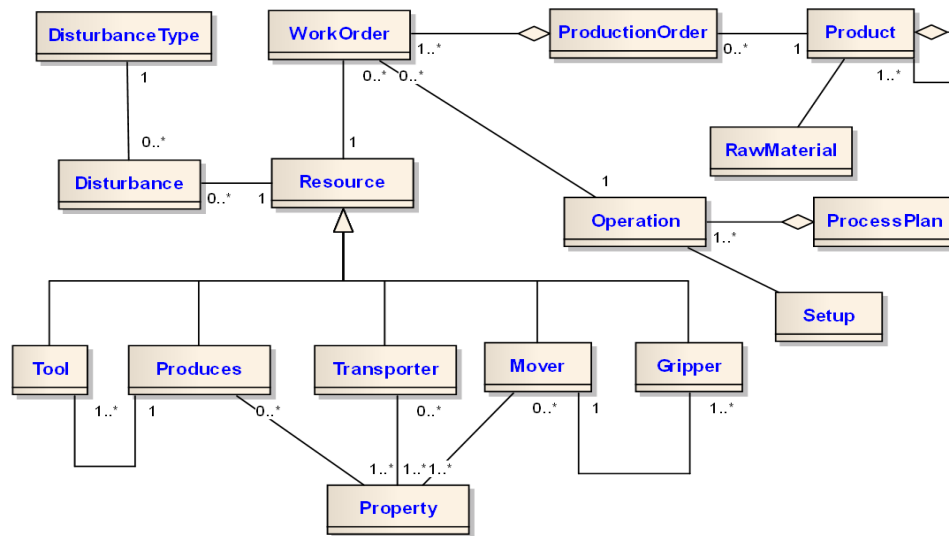


Figure 2.12: Main concepts and architecture of ADACORE Manufacturing ontology reproduced from Brogo and Leitao (2007)

2.5.2.2.4 Ontology Based Multilayer Knowledge Framework

Lee and Suh (2008) proposed an ontology-based multi layer knowledge framework (OMKF) for the PLM systems. They highlighted the limitations of present knowledge management systems with regards to the ambiguities in knowledge being shared. They introduced three types knowledge i.e. axioms, knowledge maps, and specialized knowledge for domains. They also identified four types of product models i.e. product context model, product specific model, product planning model and product manufacturing model. They used heavyweight logic to capture the semantics of concepts and knowledge at various levels of abstraction which ensured the formal capture of semantics.

The product context model of the (OMKF) formally defined several useful concepts like *Part*, *SolidGeometry*, *GeometricFeature*, *Function*, *ManufacturingProcess* and *ManufacturingMachine*. Such concepts and their formal semantics can be really useful to develop an understanding of the core concepts for the ontology being researched.

Lee and Suh (2008) also comments about important future research direction. The most relevant ones of them include (1) the need to develop product and manufacturing ontologies further for better knowledge sharing (2) the use of the rule based logical reasoning mechanism to support better reasoning capabilities over manufacturing knowledge.

2.5.2.2.5 Design for Manufacturing (DFM) ontology

The design for manufacturing (DFM) ontology by Chang et al (2010) facilitated the development of a decision support system for designer. The use of DFM ontology in the DFM enabled the analysis of the designed products for their manufacturing and also offered alternative design solutions (Chang et al. 2010). The DFM ontology was semantically sound because it used OWL and SWRL to formally capture the semantics of concepts. The DFM ontology is quite detailed with a wide range of manufacturing concepts. However, the DFM ontology was mainly focussed on the analysis with respect to the joining processes. Within the joining processes the focus was on enabling the capture of welding knowledge and analysing designed part for the weld-ability and offering alternative welding solutions. It highlighted the need to explore the ontological approach in supporting DFM in the context of other manufacturing processes like machining.

2.5.2.2.6 An Ontology for Integration of CAD and CAPP Systems

Dartigues et al (2007) developed an ontological approach to exchange product information between CAD and CAPP systems. Dartigues et al (2007) proposed a commonly understood heavyweight feature ontology that facilitated the translation of CAD and CAPP files into a neutral format that could be understood by both CAD and CAPP systems. The framework for this system as shown in figure 2.13 resembles the framework proposed for the researched ontology in section 1.2 but there are some major differences.

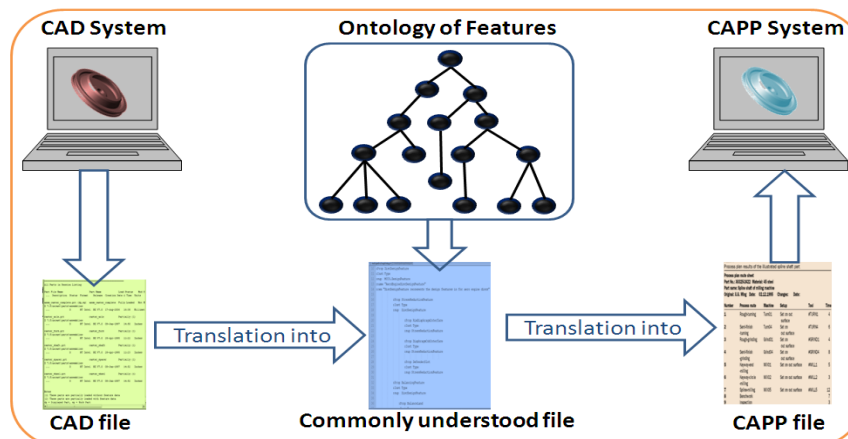


Figure 2.13: Process for data exchange between CAD and CAPP systems (Dartigues et al, 2007)

Dartigues et al's (2007) ontology and framework is only assisting the translation of CAD files into the files that are understandable for CAPP applications. Therefore, this approach does not provide the production consequences of changing the design in the design stage. Whereas, in this thesis the target is to provide the production consequences to the designers while they design parts and thus assist them in

making decisions. Dartigues et al's (2007) work does not make use of the databases to capture and share knowledge where as this thesis explores the knowledge sharing by using the common semantic base provided by a core manufacturing ontology. However, Dartigues et al's (2007) feature ontology can help to define the core concepts related to features in design and production.

2.5.2.2.7 Semantic Manufacturing Interoperability Framework

Chungoora (2010) proposed a semantic manufacturing interoperability framework (SMIF), as shown in figure 2.14, to facilitate interoperability across product design and production domains. The SMIF proposed a multilayer ontological framework where the first layer i.e. the foundation layer provided a foundational ontology for modelling the domain i.e. ontologies in second layer i.e. Domain Ontology Layer. The next two layers were defined to enable interoperability across the domain ontologies where the 'Semantic Reconciliation Layer' reconciled the concepts from the two domains for similarities and the 'Semantic Interoperability layer' enabled knowledge sharing between the two domains based on the reconciled concepts.

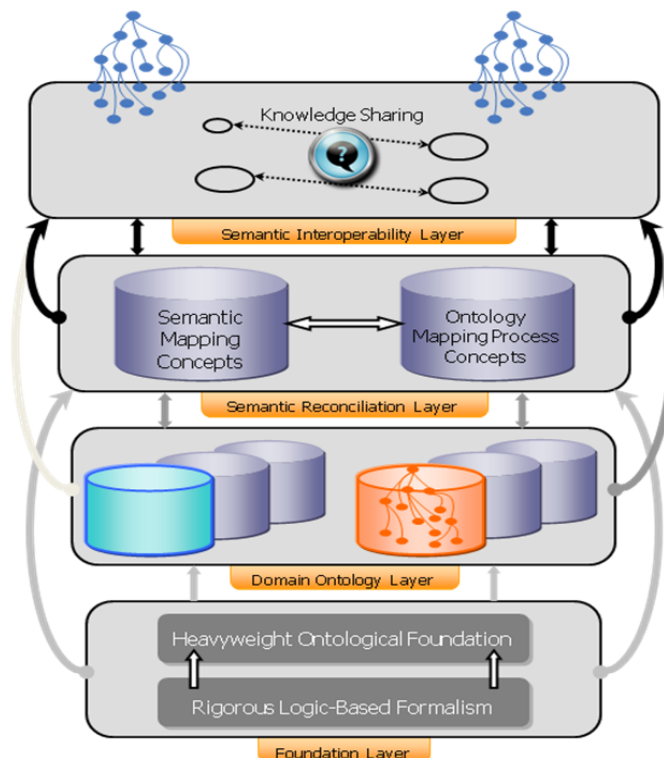


Figure 2.14: Semantic Interoperability Framework (Chungoora, 2010)

The proposed framework and ontology was successful in capturing the semantics and sharing knowledge across product design and production domains. However, SMIF was limited to support the capture of semantics and support interoperability between product design and production for simple hole features only.

The SMIF is one of the most relevant works for the research work reported in this thesis. However, the depth and range of concepts and relations need to be explored further (Chungoora, 2010) to support interoperability across a wider range of applications. The ontology developed by Chungoora (2010) is limited in terms of supporting the development of application specific product design and production ontologies. A set of concepts to support the capture of production knowledge needs to be defined. The feature and part family point of views need to be explored further to support interoperability at more detailed and specific levels (Chungoora, 2010).

2.5.3 Feature Based ontologies

Feature based design and production has been in focus for a long time (Chen and Wei 1997; Gunendran and Young 2008; Salomons et al. 1993; Wang et al. 1993). In the field of supporting interoperability through ontologies, feature based approaches have regularly been exploited by the researchers (Gunendran and Young, 2008; Abdul-Ghafour et al, 2007, Dartigues, et al 2007). A review of the feature based ontologies and ontological methods to support design and production collaboration and knowledge sharing is provided next.

2.5.3.1 Overview of Feature Based Technology

Feature based technology has two main aspects i.e. (1) feature recognition and extraction and (2) design by features. For feature recognition and extraction mathematical algorithms are used to identify and extract the geometrical features of interest. However, this approach is limited in its ability with respect to the effectiveness of the algorithms to identify related features (Martino and Giannini, 1998). This approach also has a drawback where the features can only be extracted and information about their production can only be shared once the part has been designed.

The design by features approach on the other hand can be used to share production information while designing. To share production information during design researchers have been working on an approach to develop the designs using feature from a library of manufacturing features (Chen and Wei. 1997 and Hoque and Szececi 2008). Where the design is developed using the manufacturing features, the sharing of manufacturing information to the designers is obvious.

However, this approach constrains the designers to select features from the available set of predefined manufacturing features. This effects design flexibility, functionally and accuracy because the features are defined from a manufacturing context. The approach rather than obviating the further requirement to extract and recognise

features brings the additional requirement to validating designed features according to the feature recognition methods for manufacturing (Pratt, 1993).

2.5.3.2 Feature Based View on Interoperability between Design and Production

Feature based technologies have been used to enable product information sharing between different CAD applications as well as CAD ,CAPP and KB applications (Abdul-Ghafour et al ,2007; Dartigues et al, 2007; Shah, 1995; Otto, 2001). A useful understanding of a feature is given in an ISO standard as “an area of interest on an item that is described as a unit for some purpose. A feature serves as a means of calling attention to or attaching properties to any portion of an item” (ISO 10303-207 1999).

The importance of feature as a support to link the product design and production activities has been reported extensively e.g. the group technology based approaches like PERA (PERA, 1969), Opitz classification (Opitz, 1970), Brisch system (Gallagher, 1986), MICLASS (Houtzel, 1975), DCLASS (Love, 1985) and FORCOD (Jung and Ahluwalia, 1991) to identify and relate the production methods for designed parts.

These systems can provide the generic production feedback but they cannot accommodate the complex geometries and may identify similar processes for largely different features (Holland et al, 2002). Moreover, the library of available manufacturing processes relating to the features does not represent the actual complexity involved in design and production (Holland et al, 2002).

Significant work has been done in the standards community mainly under the technical committee-184 subcommittee-4 for the exchange of product information such as the Standards for the Exchange of Product Model Data (STEP) using manufacturing features (ISO-10303-AP224 2003). The importance of feature based modelling to support integrated design and production has been recognised (Guh, 1994) and research work in this context has been continued by different researchers over different periods of time e.g. Young and Bell (1993), Han et al (2000), Aifaoui et al (2006), Ming et al (2007), and Gunendran and Young (2008).

2.5.3.3 The Use of Predefined Standard Features for Interoperability

As mentioned in the last section, use of pre defined manufacturing features for designing parts leads to compromises on design intent. Similarly, the use of pre defined design features for production will lead to compromises in production. One of the approaches that can potentially overcome these problems is to get the designers and production engineers agree on the forms of predefined features. Such features can be referred to as standard features (Usman et al, 2011). The standard feature

approach has been used to support information exchange between product design and production (Jagenberg et al, 2009). Because in this approach the features are standard across design and production, their functionality and production methods are also standardised. This offers reusability of design and production information and since the standard features have same forms across design and production the interoperability issues are resolved.

However, it is known that the different product lifecycle domains including design and production require different forms for representing their features of interest and the information attached to them (Chen and Wei, 1997, Pratt, 1993). Therefore, standard features approach can only be applied to a very limited number of design and production features (Usman et al, 2011). Enforcing the standard features approach across design and production domains can lead to compromises on functionality and production methods of parts. Moreover, the standard features approach does not ensure interoperability across many design and production features because of the different forms of design features and production features. Therefore, an approach where the design and production features can be related despite their different forms is needed.

2.5.3.4 Mapping Different Features for Interoperability

An attempt to support interoperability between product design and production systems was made by Dartigues et al (2007). Dartigues, et al (2007) proposed a feature based ontological approach to exchange product information between CAD and CAPP systems using features. Dartigues et al (2007) proposed a commonly understood heavyweight feature ontology that could act as a neutral format between CAD and CAPP system. The framework of Dartigues' et al (2007) work has already been explained in section 2.5.2.2.6. Further explanation about Dartigues' et al (2007) feature ontology and how it translates different design features into production planning features is as follows.

In order to support the translation between CAD and CAPP feature, Dartigues et al (2007) developed their feature ontology using mainly the Core Product Model (Fenves et al, 2006) and also used concepts from the international standard (ISO-10303 STEP AP48, 1992). The breakdown of the feature concepts used by Dartigues et al (2007) are shown in figure 2.15.

The differences between this approach and the proposed research approach, and the usefulness of Dartigues et al's (2007) ontology have been explained in section 2.5.2.2.6.

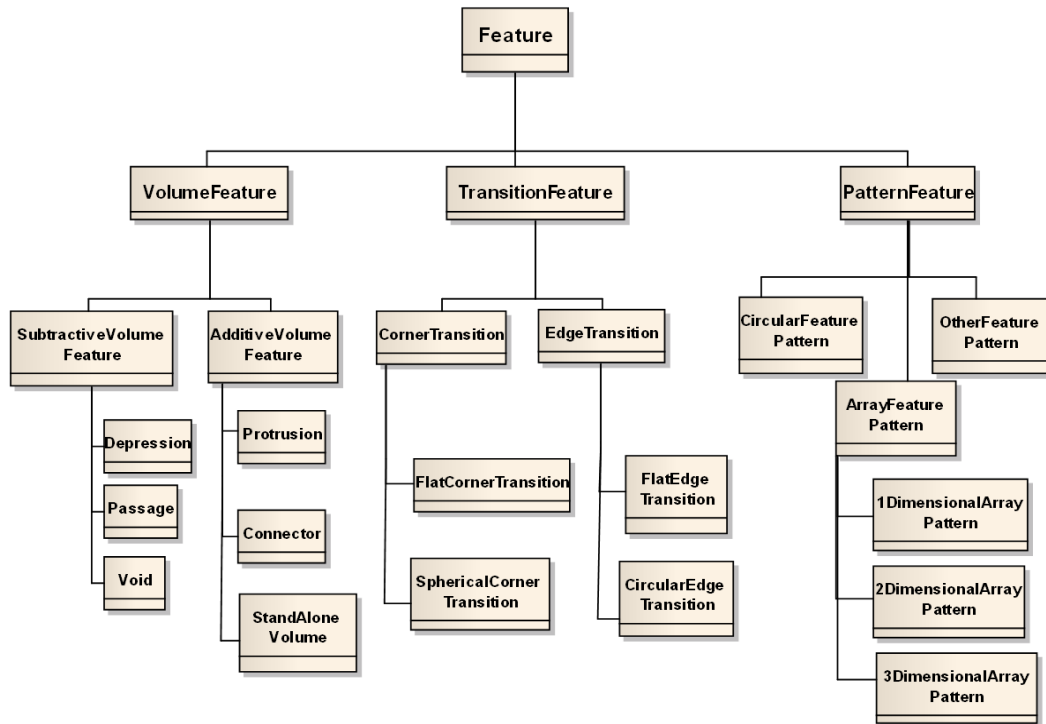


Figure 2.15: Breakdown of feature concepts (Dartigues et al, 2007)

The most relevant works in terms of sharing production knowledge with design during the design stage are Chungoora (2010), Chungoora and Young (2010) and Gunendran and Young (2010).

Gunendran and Young's (2010) work is perhaps the most relevant one in that regard. They proposed a model to support knowledge sharing between product design and production using an ontological approach based on features and part families. The most important contribution is the ability of the model to highlight production consequences of a design during the design phase. This work provides a potentially useful background input to this thesis. The formal definitions of the concepts in Gunendran and Young's model need to be captured. There is also a need to further refine the understanding of the concepts related to features. Extensions to this work are discussed in the thesis in chapters 5, 6 and 7.

Chungoora (2010) captured the formal definitions of hole features and successfully tested the interoperability across design and production domains using his heavyweight ontological framework. It was highlighted in that work that the understanding of features needed to be enhanced to support interoperability across a broader range of design and production features. Chungoora (2010) also highlighted the need to extend the heavyweight ontological foundation to support knowledge sharing across product design and production.

There are substantial differences in the methods and terminology used for design and production from a feature point of view. Features in design are defined to meet certain functional requirements whereas in production, features are defined according to the method of production (Jan de Kraker, 1998; Han and Requicha, 1995, ISO-10303-AP224 2004). A common terminology for features is not present across product design and production domains (Dartigues et al 2007). Although features are represented differently across product design and production domains, they can provide a way of relating these two domains and this aspect needs to be explored (Dartigues et al, 2007).

2.5.4 Non Manufacturing Ontologies of Direct Relevance

P. Cimiano et al, (2004) developed a an ontology named SmartSUMO as a common semantic base for developing various domain ontologies for weather forecast, match schedules, routes, hotels etc. Moreover, SmartSUMO was also designed to support interoperability across the domain ontologies. SmartSUMO can be considered as a core ontology for Smart web application domain. Therefore, this approach provides a useful understanding about the development and use of core ontologies to support the development of domain ontologies and supports interoperability across them.

A similar work was also done by Oberle et al (2007) when they developed the Smart Web Integrated Ontology (SWIntO) as a common modelling base for developing interoperable specific domain ontologies Smartweb applications.

Other acknowledged ontologies that provide potentially useful understanding of the core ontologies and their uses in supporting development of interoperable domain ontologies include the Foundational Model of Anatomy (FMA) (Rosse and Mejino Jr., 2003) and the Common Anatomy Reference Ontology (CARO) (Haendel et al., 2008) developed to suit the biomedical field.

2.6 Summary

The detailed review of the ontological frameworks, ontology development methodologies, languages and tools, and various manufacturing ontologies has been helpful in developing an understanding of the ontological knowledge systems. Particular understanding has been gained in regards to the issues faced in manufacturing knowledge systems that are using ontologies. A detailed understanding has been gained regarding the ontology based knowledge sharing between product design domains and production domains. As a result of the review, a number of research gaps have been identified as follows:

New methods to achieve interoperability through MDI and frameworks for interoperability in terms of unified and federated approaches need to be explored. Most existing systems follow the integrated approach but there is a need to understand more fully how the unified or federated approaches should be applied to support manufacturing interoperability.

From the research on ontological methods, it is clear that software systems need to understand the semantics of concepts and relations. For that purpose, heavyweight ontological approaches are required. However, the relation between foundation ontologies, core ontologies and domain ontologies and how they can be structured and exploited needs to be more clearly understood.

Over the years, many ontological methods have been developed. For the development of a manufacturing core concepts ontology, Blomqvist and Ohgren (2008) and Noy and McGuinness (2000) both appear to offer potentially useful methods when applied to manufacturing.

There is a strong focus in most of the current research on the use of OWL as a heavyweight ontological language. However, there is an issue as to whether the more expressive common logic based approaches can meet the requirements of manufacturing ontology more effectively.

In developing a manufacturing core concepts ontology there is a need to understand the extent to which concepts from researched ontologies and especially from international standards can be exploited and extended.

Feature technology should support multiple domain viewpoints and an important issue is to consider how heavyweight ontological approaches can extend the representation of feature concepts to provide increased sharing ability across feature views.

In providing a flexible approach to ontology development, it is likely that different systems will have different ontologies even if they follow a core concepts approach. This raises an issue of to how to best map and verify the concepts and relations across a number of ontologies.

The issues stated above present a wide range of research gaps in the field of ontological knowledge sharing systems. Not all of these issues, however, come under the scope of this thesis. The research gaps specifically focused in this thesis are as follows;

1. The foundation ontologies are too generic to directly support interoperability across application specific product design and production domains, therefore, what is the core manufacturing ontology that can serve this purpose.
2. The need to identify and formally define the set of concepts and relation for a core manufacturing ontology that can facilitate the development of application specific ontologies, support the capture of production knowledge and can also provide a route for knowledge sharing across them.
3. The shortcoming of present manufacturing ontologies and knowledge model when the requirement is to capture and reason about the underlying structure of knowledge need to be overcome (Correa de Silva et al. 2002).

3 An Industrial Study of Manufacturing Concepts

3.1 Introduction

In order to develop an understanding of the need for knowledge sharing and the problems hindering it from an industrial viewpoint, a twelve week industrial study was conducted in an aero engine manufacturing company. The study mainly focused on identifying the core manufacturing concepts, comprehending the relations between product design and production domains, and understanding how these concepts and relations can facilitate sharing production knowledge into product design.

It was decided in consultation with the industrial collaborators that the study would focus on one part only. The part selected was an aero engine High Pressure Compressor (HPC) disc as shown in figure 3.1.

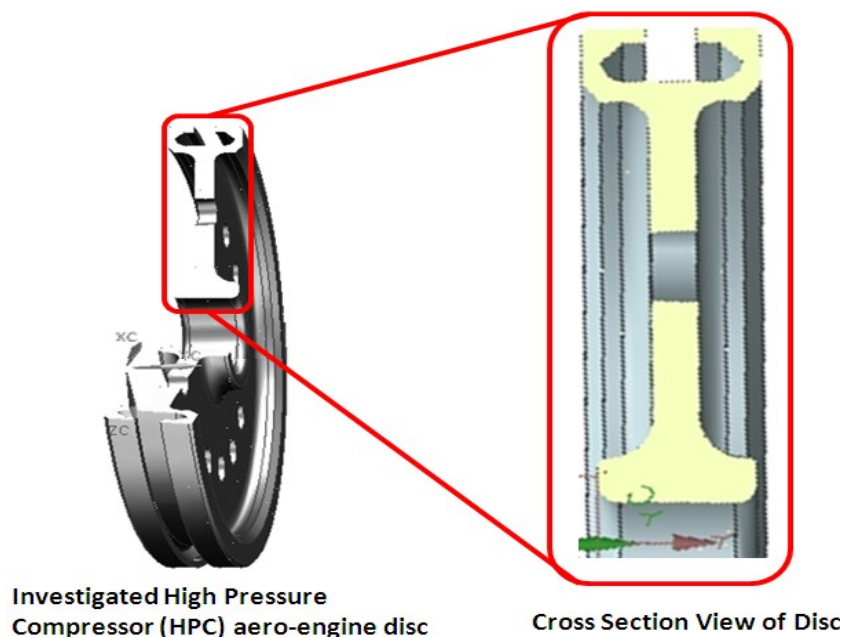


Figure 3.1: Selected engine disc for industrial investigation

This chapter reports the findings of this study. The chapter is organized as follows. Section 3.2 presents the designers' and production engineers' perspectives of the disc. Section 3.3 elaborates on the need to relate product design and production, and presents the approach identified to relate them. Section 3.4 summarizes the findings of the study.

3.2 Investigation of Disc Design and Production Systems

The first task in the study was to explore the disc design and production systems. For this purpose several meetings were conducted with various domain experts including

designers and production engineers. A comprehensive review of the documents pertaining to the design and production of the disc was also carried out. The IDEF0 methodology (IDEF0 1993) was used to model the product design and production activities, their information flows, and a detailed breakdown of these flows and activities. These models not only helped in building an understanding of the flow of information in the design and production systems, but they also proved useful in identifying the potential core concepts and relations for the proposed manufacturing core concepts ontology. These IDEF-0 models are not presented here because they were only used to develop a background understanding of the disc design and production systems, and cannot be presented to avoid the publication of any information sensitive to the company.

An important point in the industrial study was to consider the breakdown of the disc into features from design and production perspectives and to develop an understanding of those features. Another important goal was to explore the possibility of grouping similar parts into part families from both design and production points of views. Sections 3.2.1 and 3.2.2 respectively present the study of the disc from design and production perspectives.

3.2.1 Designer's Perspective of Disc

Principally the product designers were interested in the functional requirements of the disc and how these requirements can be satisfied. An investigation of the disc features and grouping discs into part families with respect to the functional requirements was carried out. The next section presents an investigation of division of the disc into features with respect to the functional requirements.

3.2.1.1 Exploration of Disc Design Features

It was found that product designers can view the disc by dividing it into different features having different forms to satisfy specific functional requirements. After developing preliminary models of disc design, the designers were interested in those regions of the disc which were critical when exposed to various thermal and working stress conditions. Thermal and stress requirements are also functional requirements which affect the division of the disc into features. A design view of the disc (figure 3.2) with respect to all its functional requirements and the form features addressing those requirements was developed as a result of the study.

The disc was divided into twelve design features with respect to their major intended functions. A feature-functionality matrix was developed to rate the importance of each

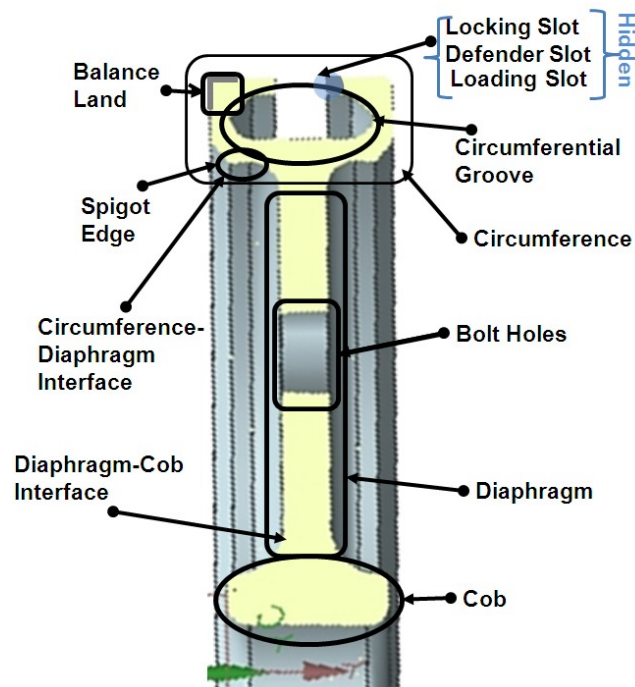


Figure 3.2: An adaptation of design perspectives of the studied disc

feature with respect to a particular function as shown in table 3.1. It was found that each feature was designed to satisfy one or more main functions. However, that feature could also satisfy some other functions as well. This is illustrated by allotting scores to features against their functions in table 3.1. A score of '9' shows the primary function for which a feature is designed and a score of '5' and '1' shows the secondary and ternary functions respectively supported by a feature.

Table 3.1: Design Feature functionality matrix

Design Features Function		Circumference	Circumferential Groove	Spigot Edge	Diaphragm	Cob	Balance Land	Circumference Diaphragm Interface	Diaphragm Cob Interface	Bolt Hole	Loading Slot	Defender Slot	Locking Slot
1	Transmit Torque	9			5	1				5			
2	Join With Other Discs			5	1					9			
3	Position Blades Axially	1	9		1								9
4	Position Blades Radially	1	9	5						5			
5	Reduce Stress Concentration							9	9			9	
6	Reduce Disc Vibration						9						
7	Reduce Stress for Loading Slot											9	
8	Maintain Concentricity with other Discs	1	5	9									
9	Reduce Shear Stress on Bolts			5									
10	Balance the Radial Forces and Stresses					9							
11	Load Blades into Circumferential Groove										9		
12	Connect Cob and Circumference				9								

3.2.1.2 Exploration of Disc Design Part Families

It was observed during the industrial study that different discs with similar functional requirements could be grouped into part families. For example, the main functional requirement of discs in an aero engine compressor was to compress the air and withstand the high pressures generated as a result. Therefore, the part families of discs in an aero engine compressor with respect to the pressure they withstand can be described as follows;

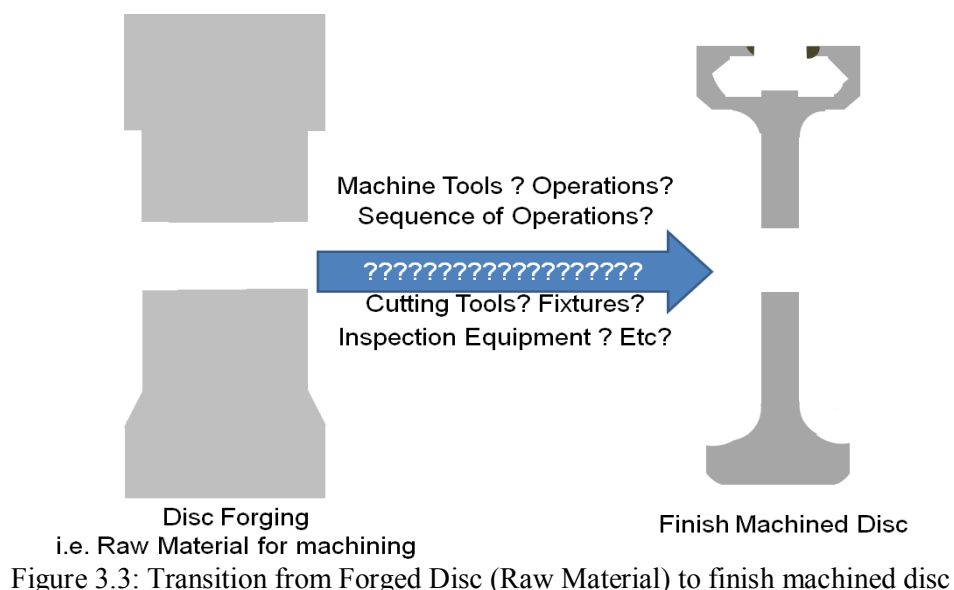
1. High Pressure Compressor (HPC) Disc Family
2. Medium Pressure Compressor (MPC) Disc Family
3. Low Pressure Compressor (LPC) Disc Family

Similarly, there can be other disc design part families e.g. High Pressure Turbine Disc Family and Low Pressure Turbine Disc Family. However, the study was primarily focused on one disc only and therefore a more comprehensive understanding of design part families was beyond the scope of this industrial study.

Since the research work reported in this thesis is targeted on the capture and sharing of production knowledge, the study focused more on the disc production system and the next section reports this portion of the industrial investigation.

3.2.2 Production Engineer's Perspective of the Disc

It was observed that the communication between designers and production engineers was mainly through meetings, and exchange of printed documents. There was no computational system to feedback the production knowledge to the designers. It was also observed during the industrial investigation that the main input from designers into the production process was an engineering drawing with a three dimensional model of the required disc. The raw material for the production of the disc was provided in the form a forged disc as shown in figure 3.3.



It was also found that important production knowledge was contained in the production method of the studied disc and its features. A production method for a disc is called 'Process Plan' which consists of a sequence of operations. A Process Plan contained information from several different areas like manufacturing resources, manufacturing facilities, and manufacturing processes.

3.2.2.1 Knowledge Abstraction in Disc Production Methods

Before developing a detailed process plan, production engineers were interested in identifying the kinds of required machine tools, fixtures, and a general sequence of operations for the production of a disc. This was required to find out whether the disc was manufacturable within the available facilities, resources, and production methods. In order to do that, a generalized or abstract view of machine tools, fixtures, and operations was required. Similarly, before generating detailed process plans, an abstract process plan representing the detailed process plans was required.

Figure 3.4a shows an abstract view of required operations, their order, and machines which are capable of carrying out those operations. Each coloured portion in figure 3.4a represents a generic view of a particular feature of the disc being manufactured using a particular method during the production of the disc.

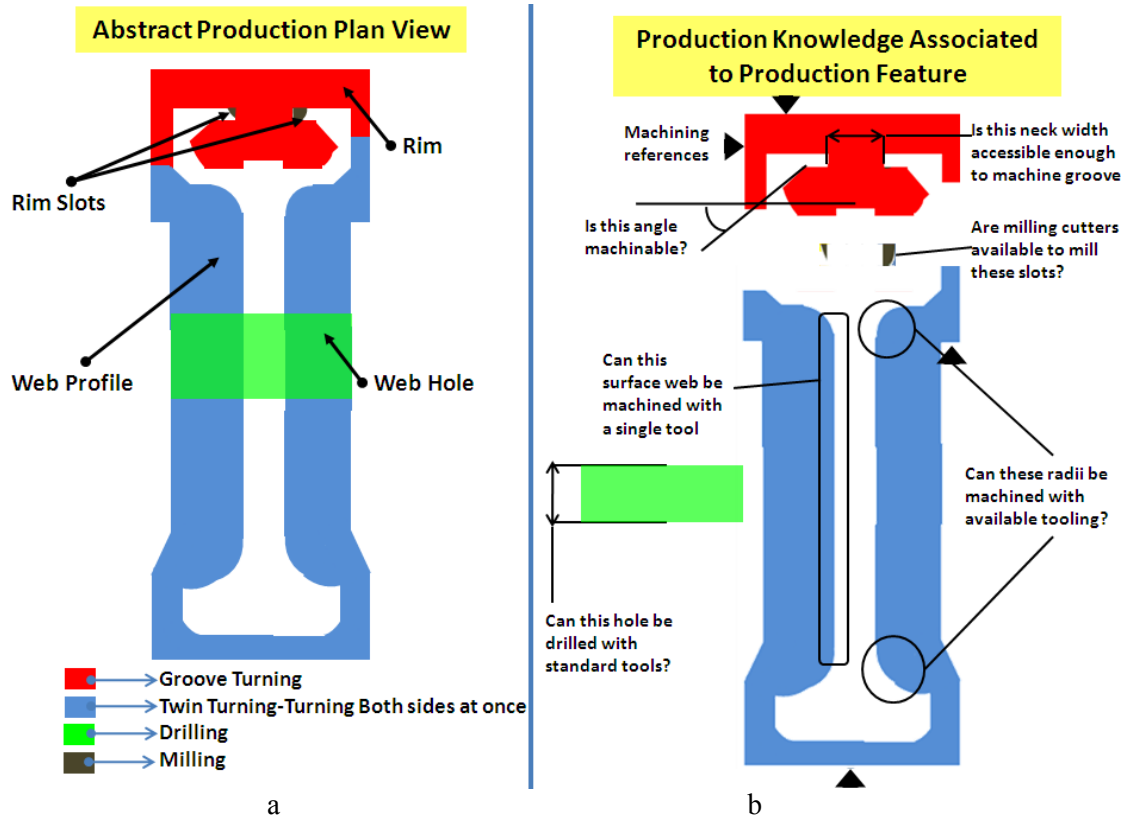






Figure 3.4: An adaptation of production perspectives of HPC disc

3.2.2.2 Exploration of Disc Production Features

From a production perspective, the disc can clearly be divided into features with respect to the method of production used for creating particular forms. These features are called production features. For example, the disc under consideration can be divided into the production features as shown in table 3.2.

Table 3.2: Matrix for Production Features vs their production method

Production Features Production Method	Rim 	Web Profile 	Web Hole 	Rim Slots 
OP-30-Rough Turning	<input checked="" type="checkbox"/>			
OP-50-Rough Twin Turning		<input checked="" type="checkbox"/>		
OP-70-Finish Turning	<input checked="" type="checkbox"/>			
OP-80-Finish Twin Turning		<input checked="" type="checkbox"/>		
OP-100-Drilling			<input checked="" type="checkbox"/>	
OP-110-Milling				<input checked="" type="checkbox"/>
OP-120-Balance Turning	<input checked="" type="checkbox"/>			

These production features are represented by different colours in figure 3.4 and table 3.2. The production features are not the geometrical portions of the disc but the forms of material to be removed from the forged disc. Each production feature has been associated with its method of production by the '☑' mark as shown in table 3.2. As shown in table 3.2 the method of production of a feature can consist of multiple yet similar operations. For example, the 'Rim' feature was found to be produced in three turning operations. These operations are 'Rough Turning', followed by 'Finish Turning' and finally 'Balance Turning'.

The capture of abstract production methods for the disc and its production features can enable the capture of important production knowledge associated with those features. For example, consider the 'Rim' feature and its production method. The production method of this feature can be extracted from table 3.2 as shown on the left hand side in figure 3.5 which lists the operations involved in the production of 'Rim'. Examples of questions that can help to identify important production knowledge associated with the 'Rim' are, "Is the neck width of Rim's groove wide enough to allow access for a cutting tool for machining?" and "Can the Rim's groove angle be machined with the standard tools available?"

It was identified during the industrial investigation that production knowledge could be associated with the operations and hence linked to production features and design features. For example, for the production feature 'Rim', production knowledge can be captured as shown on the right hand side in figure 3.5. Figure 3.5 also shows how this knowledge is associated to the production method of 'Rim'. The knowledge captured in the rules can answer the questions raised in the last paragraph and thus this approach

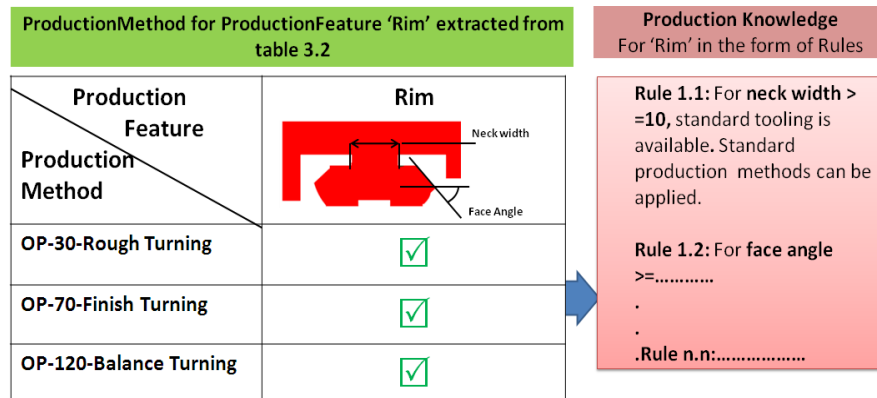


Figure 3.5: Example of extracting the production method of features and capturing their production knowledge

can facilitate the capture and sharing of production knowledge associated with features.

The above discussion shows that the industrial study has helped in developing an understanding of the way the production features can be defined, and the way their production methods and the knowledge associated with those methods can be captured.

3.2.2.3 Exploration of Disc Production Part Families

Similar to the design part families, it was found that discs could also be grouped into part families with respect to their production methods. The understanding of a production part family is based on the conceptualization of part families presented by Gunendran and Young (2010). Thus, a production part family can be constituted by grouping discs with respect to similar process plans. The similarity of process plans is based on the similar operation sequence and the use of similar machines and fixtures. This implies that a process plan for a specific disc part family can be applied for the production of any of the discs in that part family with the addition of some specific details for each individual disc. Some of the suggested part families of disc from a production perspective are;

1. Stand Alone Disc Part Family: The company used the term 'Stand Alone Disc' to refer to discs which were symmetrical on both sides and which were not welded with any other disc. The web profile of discs belonging to the family of standalone discs can be machined simultaneously from both sides through a twin turning method on a twin turning machine.
2. Projected Disc Part Family: Represents discs having projections on their web profile. The web profile of these discs cannot be machined simultaneously from both sides. Thus, the twin turning method is not applicable to the Projected Disc Family.

Once the knowledge about the features and the production of the disc is captured, the production engineers are able to determine if the disc is manufacturable with the available resources and methods. The production consequences of a change in the design of a particular feature or the inclusion of a new feature in a part family can also be determined. It is also possible to query the manufacturability of features within the part family production methods. However, in order to understand the production consequences of a change in the design of a feature or the addition of a new feature to a part family, the relations between the product design domain and production domain needs to be understood. The next section discusses the relations between these two domains in detail.

3.3 Relations between Design and Production

3.3.1 Need to Relate Product Design and Production Concepts

One of the potential ways explored in industry to relate product design and production domains was the standardization of features across disc design and production. This meant that a standardized feature would consist of a standardized form across product design and production. The standard feature is therefore, intended to satisfy a standard functional requirement and will have a standard production method. This further implies that a standard feature will simultaneously be a production feature and a design feature. This way, the production knowledge associated with a standard feature can automatically be shared with designers. However, it was identified that most of the features on a disc cannot be standardized. This is because the forms of the features important to designers are different from the forms of the features important to production engineers. Moreover, because of the differences in designers' and

production engineers' perspectives, the concepts used by them and their semantics were also different. Figure 3.6 provides a view of the mismatches between design and production perspectives of the studied disc.

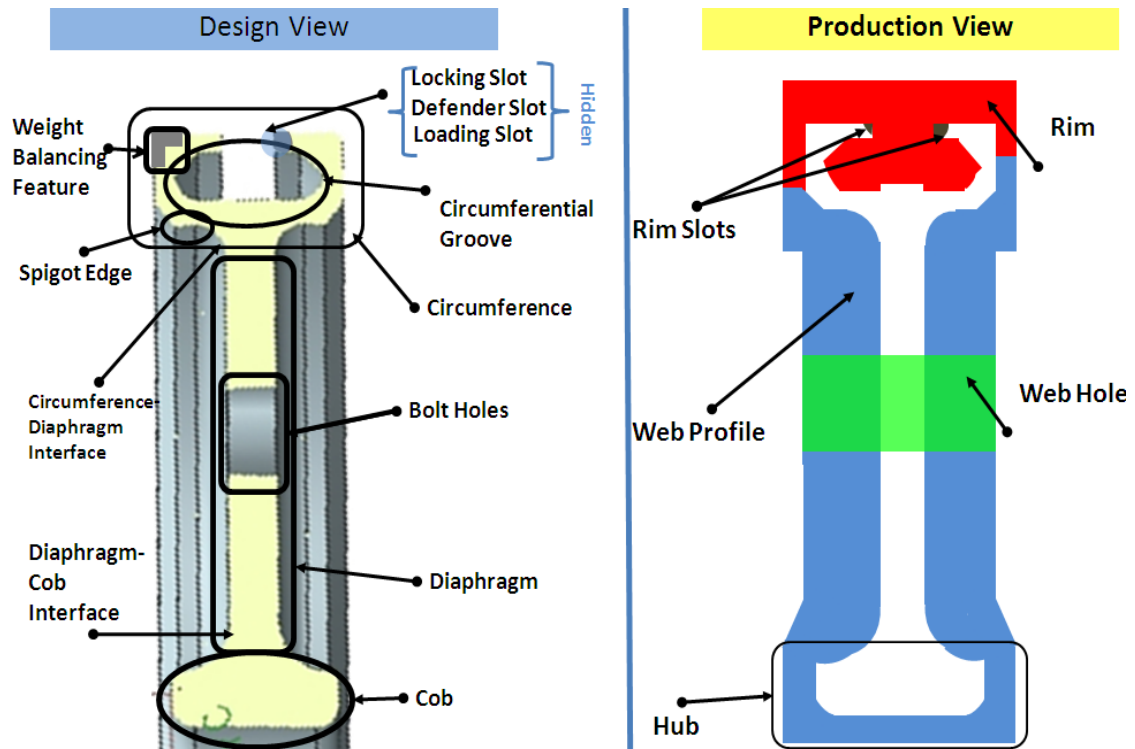


Figure 3.6: Mismatches between production design and production concepts

During the industrial investigation two types of mismatches were identified between product design and production concepts. The first type of mismatch was a term mismatch occurring due to designers and production engineers referring to the same forms on a disc using different terms. For example, a hole in the disc was termed 'Bolt Hole' by the designers and 'Web Hole' by production engineers as shown in figure 3.6. Another example shown in figure 3.6 is Cob versus Hub.

The second type of mismatch occurs due to the difference in the semantics of terms used in product design and production domains. As discussed in section 3.2.1, designers defined features with a view to meet certain functional requirements. While, as discussed in section 3.2.2, production engineers define features with respect to their method of production. For this reason the intended meanings or semantics of concepts were also found to be different. For example, the production feature 'Web Hole' as shown in figure 3.6, is defined with respect to the method of production which is 'Drilling'. The design feature 'Bolt Hole' which has the same form as that of 'Web Hole' is designed to meet the functional requirements of joining the studied disc with other

discs using bolts. This indicates that even when the designer and the production engineers refer to the same feature, they may not only use different terms, but they may also associate different semantics with them.

Other examples are more complex where design and production features do not even refer to the same portion of the disc. For instance, the production feature 'Rim' as shown in figure 3.6, encompasses four different design features i.e. Locking Slot, Loading Slot, Balance Land, and Circumferential Groove. Similarly the production feature 'Web' covers five design features i.e. 'Diaphragm', 'Diaphragm-Cob interface', 'Circumference-Diaphragm Interface', 'Spigot Edge' and 'Cob' as shown in figure 3.6. This shows that, the relations between features from these domains can be, 'one to many', 'many to one' and 'one to one'. In such cases, sharing of the production knowledge with product design becomes a complex issue. This highlights the need to find a way to relate the design and production features. The next section elaborates this aspect of the industrial study.

3.3.2 Relating Design and Production for Knowledge Sharing

It is understood that design features have an associated function and production features have an associated production method. It is also understood that both design and production features have forms. In order to relate the features from design and production domains, some commonalities need to be identified between them.

As a result of the study, the overlapping portions of the forms of design and production features can provide the common basis for relating design and production features of the disc. However, it is understood that the forms of design and production feature are different. But, It is also known from the last section that the form of a production feature can encompass forms of one or more design feature and vice versa. Thus, in order to know the production consequences of changing the design of a design feature, the knowledge system needs to identify those production feature(s) that encompass the form of design feature under consideration. Once a link is established through this method, production knowledge associated to the relevant production feature(s) can be shared with designers through that link. This approach is explained in the next section with the help of an example taken from the industrial investigation.

3.3.2.1 Example: Relating Design & Production Features for Knowledge Sharing

Consider a case where the production consequences of changing the value of neck width of a design feature named 'Circumferential Groove' are required to be found. The production consequences can be found from the production knowledge associated with the production feature corresponding to 'Circumferential Groove'.

One way of doing this is to first identify that production feature(s) that encompasses the design feature in question. In this case the production feature named 'Rim' encompasses the design feature 'Circumferential Groove' as shown in figure 3.7. Thus, the production knowledge associated to 'Rim' is relevant for the design feature 'Circumferential Groove'. This means that 'Rim' and 'Circumferential Groove' are corresponding concepts from production domain and design domain respectively. This identified 'Rim' as the production feature corresponding to the design feature 'Circumferential Groove'. Therefore, the production knowledge associated with 'Rim' is relevant for the design of 'Circumferential Groove'.

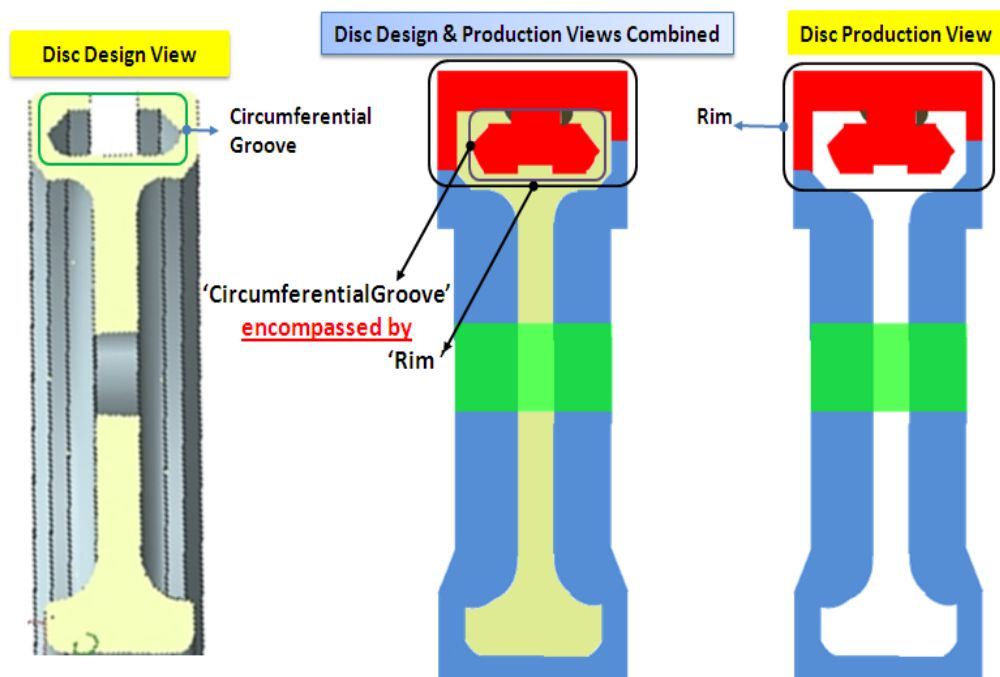


Figure 3.7: Finding Production Features encompassing the form of design feature 'Circumferential Groove'

From section 3.2.2 of this chapter, it is understood that production knowledge can be related to the production method of a production feature. Following this approach and based on the established relation between 'Circumferential Groove' and 'Rim' a method can be devised for sharing production knowledge with designers. An example of this method is shown in figure 3.8.

Figure 3.8 shows the design view of 'CircumferentialGroove' along with its function and parameters shown on the left hand side. The figure shows the production feature 'Rim' with its production method and the associated production knowledge on the right hand side. In this example, designer needs to know the production consequences of changing the neck width of 'CircumferentialGroove'. In this case, the production

knowledge associated with the operations in which neck width is produced is of relevance to the designer. The rule depicting the knowledge is shown in figure 3.8 as 'Rule1.1'. This rule states "For a **neck width** size range of **10mm to 12mm**, standard tooling is available **and** standard machining methods can be applied". This knowledge can be fed back to the designer through the established link shown by the arrow going from production view to the design view in figure 3.8. This can then assist the designer

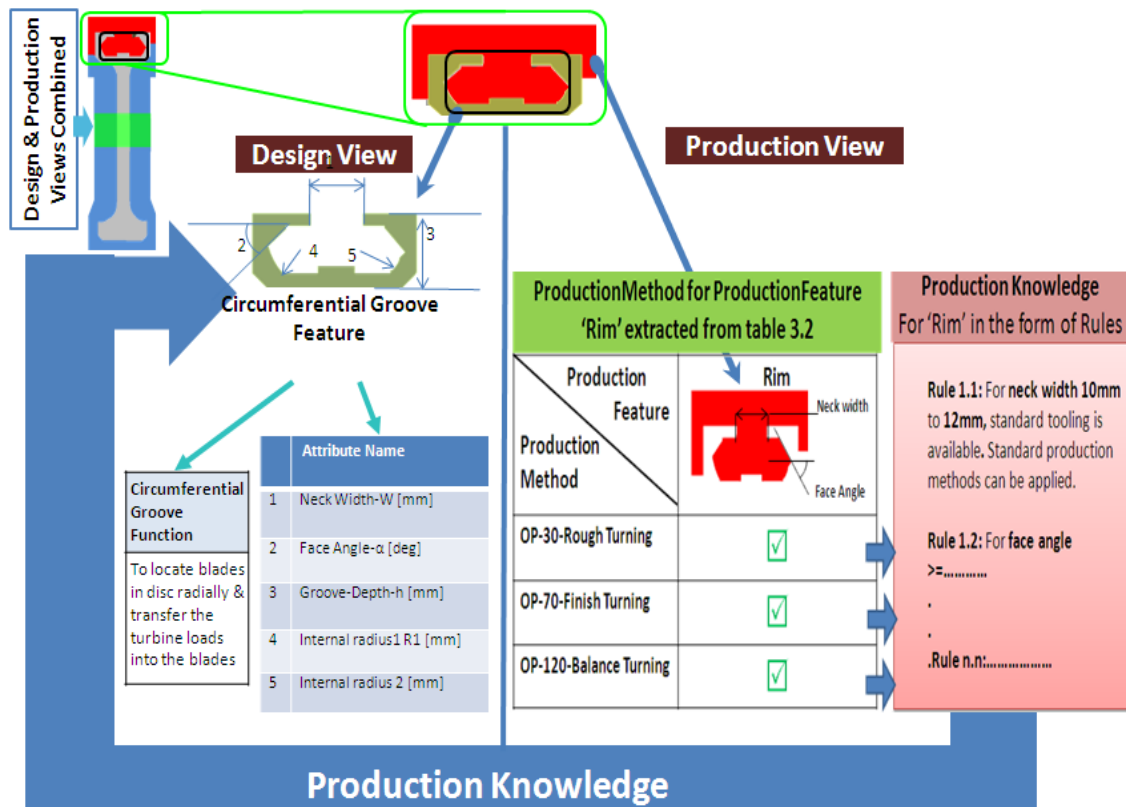


Figure 3.8: Mechanism for sharing Production Knowledge into design through features in making appropriate decisions about the design of 'Circumferential Groove' in the light of its production consequences. This approach has been used as the basis for mapping between the design and production features. Through the same approach, knowledge associated with other features can also be captured and shared.

3.4 Summary

The industrial investigation has helped in understanding the idea that underlies the development of core concepts. The study gives an industrial perspective of core concepts that are required and has helped in understanding the core issues when developing a manufacturing ontology.

The industrial study helped in developing an understanding of both the product design and production perspectives and their relations. The differences in the product design

and production perspectives and the need to relate these domains are clarified through this study. The study also assisted in understanding how product design and production can be related to each other to facilitate the sharing of production knowledge for product design. The study highlighted the need for product design and production views to be based on a common understanding. This is needed to support knowledge sharing between product design and production domains.

It was also established during the industrial investigation that the production knowledge exists at multiple levels of abstraction. This finding highlights the need to capture and share knowledge at these levels. This is required because production engineers need to know the abstract process plans and feature production methods during early phases of production planning before generating the detailed process plans.

The study has aided the understanding necessary to define the core concepts. For instance, a comprehensive understanding of the concepts *DesignFeature* and *ProductionFeature* have been gained. An understanding of other concepts i.e. *ProcessPlan*, *Operation*, *ManufacturedPart* and its various states, *ProductionMethod*, and *PartFamily* have been gained. The exploration of core concepts and relations from the literature and the industrial study is discussed in detail in chapter 5.

4 The Requirements of a Manufacturing Core Concepts Ontology

4.1 Introduction

This chapter outlines the requirements of a manufacturing core concepts ontology and outlines the research questions identified regarding them and also introduces the novel aspects of the manufacturing core concepts ontology.

The chapter is organized as follows. Section 4.2 points to the general issues identified. Sections 4.3, 4.4 and 4.5 discuss the main research issues and outline the research questions against which the contribution to knowledge is made. Section 4.6 presents a summary of the chapter.

4.2 Research Issues in Manufacturing Ontologies

Within the area of ontological manufacturing knowledge systems extensive research work has been undertaken as reported in sections 2.4 and 2.5. However, the present knowledge systems fall far short of the requirements of modern manufacturing industry (Fischer and Stokic 2002). This is because the present manufacturing ontologies do not all essentially address the knowledge sharing requirements in product design and production (Chungoora et al. 2010). Moreover, these ontologies have varying expressiveness (Ray 2004) and varying capability to interact with other ontologies. Therefore, the seamless exchange of product design and production semantics to support knowledge sharing has still not been achieved (Chungoora 2010).

Based on the initial investigation of the research problem through the literature (chapters 2) and the industrial study (chapter 3), a wide range of research issues can be identified. These issues relate to the integration of AI technologies with manufacturing systems, mapping and verification of ontologies, developing ontologies for worldwide supply chains, and building of trust between interoperating partners. Not all of these issues, however, come under the scope of this thesis, therefore, only issues relevant to this thesis are listed below:

4. Research work can be directed to simplify the complexities involved in reusing existing ontologies for engineering new ones (Fischer and Stokic 2002).
5. Research work can be aimed at addressing the issues regarding ontologies and semantic interoperability to assist knowledge sharing between product design and production (Chungoora and Young 2010; Lin et al. 2004)

6. There exists a research gap for the provision of generally agreed and explicitly defined underlying concepts within the manufacturing world. The present manufacturing ontologies need to be completed, refined and developed further (Chen et al. 2008; Chungoora 2010; Lee et al. 2005).
7. A heavyweight ontological foundation for product design and production needs to be explored for offering a richer semantic base with more expressiveness, better knowledge capturing capability, and enhanced inference power (Chungoora et al. 2010; Lin et al. 2004; Lin 2007).
8. Researchers should explore the possibility of defining multiple levels of foundation for the manufacturing domain (Young et al. 2007).
9. The features and part family context for sharing manufacturing knowledge with product design using ontologies needs to be explored further (Abdul-Ghafour et al. 2011; Chungoora and Young 2010; Dartigues et al. 2007).
10. The shortcoming of ontologies when the requirement is to share the inferences and underlying structure of knowledge need to be overcome (Correa de Silva et al. 2002).
11. The author of this thesis believes that researchers need to explore the ways to capture and reason about not only the actual asserted production knowledge but also the structure of that knowledge.

Point 1 regards the complexities in reusing existing ontologies for developing new ones. In this thesis concepts from existing manufacturing ontologies are reused but the main research focus is not on solving the problems in reusing them.

Points 2-8 focus on the primary issues related to the identification of concepts and relations for manufacturing ontology, explicitly defining those concepts and relations, utilizing them to support knowledge sharing across product design and production and the capture and reasoning about knowledge as well as the structure of knowledge. This thesis puts forward the potential solutions to these issues.

The novel aspect of the proposed solution is to define a set of key manufacturing concepts which can provide a commonly understood formal semantic base which can support (1) the development of application specific product design and production ontologies, (2) the capture of production knowledge and (3) knowledge sharing between product design domain and production domain. This aspect is discussed in section 4.3.

A further aspect regards the capture of varying depths of meaning of concepts from the generic level to the specific product design and production levels. This aspect is explained and discussed in section 4.4.

Another aspect regards the capture and sharing of manufacturing knowledge at multiple levels of abstraction as explained briefly in section 3.2.2. This aspect is elaborated and discussed in detail in section 4.5.

4.3 Identifying Concepts & Relations and Formalising their Semantics

It is understood that domain ontologies (defined in section 2.4.2.2.3) are typically developed for a specific domain e.g. product design and production are two specific domains. If the meanings of the concepts are commonly understood by the domain community, there is less need to formally capture the semantics. Therefore, the domain ontology can be restricted to a lightweight formalization only.

On the other hand, the meanings or semantics of concepts in autonomous lightweight domain ontologies lack formal rigor. The loose formal semantics can result in ambiguities in formal definitions of concepts. This means that concepts in lightweight ontologies are open to multiple and possibly inappropriate interpretations. The consequences of that are interoperability problems.

With the semantics not captured in formal logic, software systems cannot accurately perceive either the differences or the similarities between design and production concepts (Oberle et al. 2007). This highlights the need to capture the semantics of concepts in formal logic i.e. the need to develop heavyweight product design and production ontologies.

However, if the heavyweight product design and production ontologies are developed independently, the interoperability across them can still remain very limited. This is because product design and production domains require different concepts with different semantics that are not clearly understood across the two domains as discussed in section 3.3.1. Therefore, a common set of concepts need to be defined for ensuring interoperability between product design and production domains. If a common set of concepts defined in formal logic underlies the specific product design and production ontologies, the system can understand the meanings of similar concepts and identify semantically different concepts. The nature and scope of these common set of concepts, however, need some deliberation.

Foundation ontologies (defined in section 2.4.2.2.1) can provide the required common set of formally and unambiguously defined concepts. Foundation concepts are

developed and defined with a view to encompass the basic semantics of everything which makes these concepts generic enough to be agreed upon by all domains. The foundation concepts are useful at the generic level, however, they are overly generic and wide-ranging (Borgo and Leitão 2007) to be used for effective knowledge sharing between product design and production domains.

For instance, the foundation concept '*Resource*' may refer to *Machine Tools, Cutting Tools, Operators, CAD-Software Tools, Printers, Time, and Space etc.* This implies that a huge set of concepts which are semantically different may well be classed under a single foundation concept. A huge set of concepts classed as *Resource* would lead to the identification of similarities between vastly different concepts when viewed from the level of product design and production. This can create ambiguities for knowledge sharing across specific product design and production domains. So, a more specialized set of concepts with their semantics closer to the product design and production domains are required to support knowledge sharing between them.

To highlight the requirements, there is a need to define an intermediate set of concepts between the foundation ontologies and the application specific product design and production ontologies as shown in figure 4.1. Such intermediate level concepts and relations should be neither as generic as foundation ontologies nor as

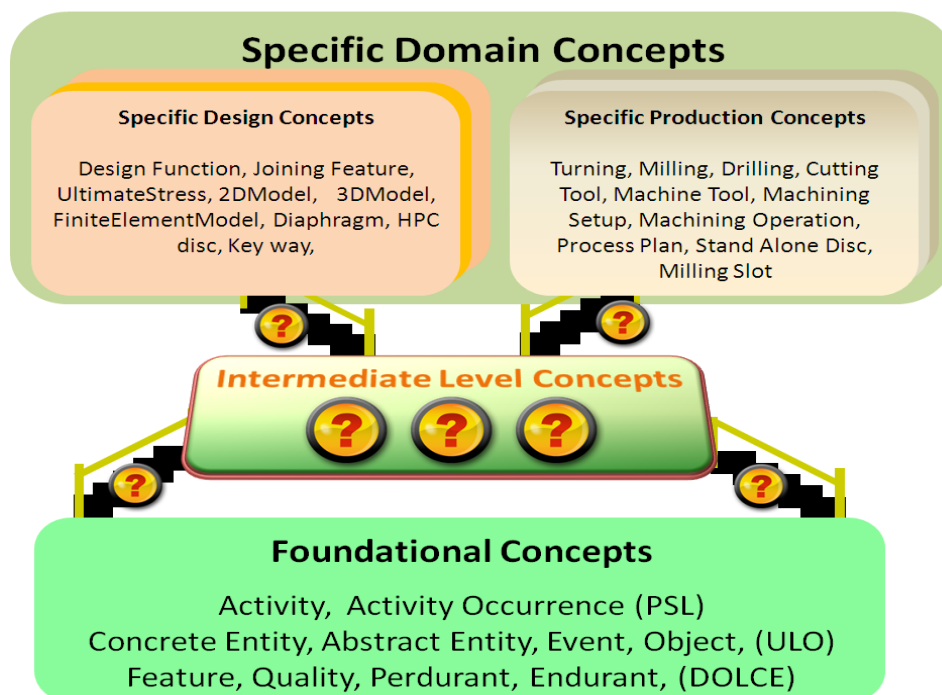


Figure 4.1: The intermediate set of concepts between foundational and domain specific concepts
specific as domain ontologies. Moreover, such concepts and relations should be to

able assist in the capture and sharing of production knowledge into product design. The following research questions arise in this regard.

1. What are the intermediate or core set of manufacturing concepts and relations that sit between the foundation and application specific domain ontologies?
2. Can these core concepts and relations support the development of application specific product design and production ontologies?
3. Can these concepts and relations provide a route for knowledge sharing between product design and production domains for relating the two domains?

Moreover, this common set of concepts and relations must be defined formally which raises the following research question.

4. Can the semantics of the core concepts and relations be captured formally using heavyweight logic so that the knowledge system can computationally understand the meanings of concepts and thus eliminate the ambiguities in their interpretations? Can this help identify the similarities between concepts from product design and production domains.

Researchers have identified many different classifications of concepts and relations within the manufacturing area. Key references in this regard are; the classification of manufacturing resource (Leimagnan et al. 2006; Vichare et al. 2009), classification of manufacturing processes (Feng and Song 2003; ISO-18629-1 2004; Todd 1994), classification of manufacturing facilities (Lin and Harding 2007; Simpson et al. 1982; Zhao et al. 1999), and features and part families models for relating manufacturing to product design (Chungoora and Young 2010; Gunendran and Young 2008; Young et al. 2007).

There are several other relevant pieces of work on ontologies and ontological approaches that have been mentioned in section 2.5. The most important ones in the context of this work are the product process integration model (Martin and D'Acunto 2003), model of a flexible manufacturing facility (Molina and Bell R 1999), MASON ontology (Leimagnan et al. 2006), MSE ontology model (Lin and Harding 2007; Lin et al. 2004), CIMOSA architecture (Kosanke et al. 1999), FDM model (Harding and Yu 1999), Process Specification Language PSL (ISO-18629-1 2004), Core Product Model by NIST (Fenves et al. 2006), ADACHORE core manufacturing ontology (Borgo and Leitão 2007), ISO-10303 STEP AP-1, AP-224, AP239, AP-49 and other relevant APs, Requirements for establishing manufacturing enterprise process

interoperability (ISO/CEN-11354 2008), Common Logic standard ISO/IEC-24707, and Industrial Manufacturing Management Data (MANDATE ISO-15531 2006).

These models, ontologies and standards can potentially contribute to the core set of manufacturing concepts and relations. The first step in the development of manufacturing core concepts and relations is to synthesize the understanding gained from the above mentioned research works into a commonly understood manufacturing ontology for supporting knowledge sharing between product design and production domains. Questions 1 and 4 are addressed in chapter 5 of this thesis which defines the core set of manufacturing concepts and relations and presents the formalisation of their semantics in formal logic. Questions 2 and 3 are dealt with in chapter 6 of this thesis.

4.4 Issue of Capturing Varying Depths of Meaning of Concepts

The idea of core concepts and relation i.e. core ontology is not new (Gangemi and Borgo 2004), however, it has not been applied to support knowledge sharing between product design and production domains. Normally, core ontologies are developed using foundation ontologies to take advantage of the formal basis provided by them (Borgo and Leitão 2007). It has also been understood that the core manufacturing ontology should support the development of specific product design and production ontologies. This shows that concepts in the core manufacturing ontology will have varying depths of meaning as some of them will have generic meanings whilst other will be closer to the product design and the production domains. The variation in the depths of meaning of manufacturing concepts can be further elaborated by revisiting the concepts and relations found during the literature review in chapter 2 and the industrial study in chapter 3.

During the literature review and the industrial exploration of product design and production domains, a large number of concepts were found. For example; Particular, Endurant, Perdurant, Inspection, Batch Card, Cutting Tool, Setup, Fixture, Quality, Concession, Critical Feature, Standard Feature, Production Feature, Design Feature, Dimension, Tolerance, Function, Stress, Machine Tool, Turning, Twin Turning, Rim, Diaphragm, Part Program, Engineering Drawing, WorkPiece Material, Tool Material, Functional Design Info, Engine, Turbine, Compressor, Blade, Circumferential Groove, Fir-tree Slot, 2DFeature, Milling, Washing, Object, Change Request, etc. These concepts were found to be either

- Applicable to any domain e.g. *Feature*, *Resource* and *Event* or

- Applicable to any of the product lifecycle domains *Part*, *Product* and *PartFamily* or
- Specific to the individual manufacturing domains. For instance *DesignFeature*, *DesignFunction* and *DesignPartFamily* for product design domain and *ProductionFeature*, *ProductionPartFamily*, *MachineTool*, and *CuttingTool* for the production domain.

This shows that there are manufacturing concepts having varying depths of meaning from being generic to being very specific. In addition to different manufacturing concepts having different depths of meaning, a single concept can also have varying depths of meaning with respect to its application areas. This is illustrated in figure 4.2.



Figure 4.2: A view of “Resource” concept and its varying interpretations

Figure 4.2 depicts the various possible interpretations of the concept ‘*Resource*’. *Resource* can be interpreted as a very generic concept which is not specific to any domain. At a more specific level *Resource* can be applicable only to the product lifecycle e.g. *PartMaterial*. Similarly, *Resource* can be interpreted to be specific to a design domain e.g. *CAD-System* and to a production domain e.g. *CuttingTool* and *MachineTool*.

This shows that the concepts in the core manufacturing ontology will gradually evolve from generic to more specific levels with respect to the depths in their meanings. The more specific concepts should support the development of specific product design

and production ontologies. This highlights the requirement to capture the variations in depths of meaning of concepts from generic levels to the product design and production levels. This requirement raises the following research question.

1. How can the varying depths of meaning of manufacturing concepts, from generic to the specific levels, be effectively and formally captured?

The proposed solution for this research question is reported in chapter 6.

4.5 Capturing and Reasoning at Multiple Levels of Knowledge Abstraction

Before explaining the requirements to capture knowledge at multiple levels of abstraction it is important to explain the methods that can be used to model the different levels of knowledge abstraction.

4.5.1 Knowledge Abstraction Levels: The Meta and Instance Relations

A typical approach to knowledge modelling involves defining a structure composed of 'classes' which provides the basis for capturing the actual knowledge. Classes represent the abstractions of actual objects and events. In ontological engineering 'classes' can be considered equivalent to 'concepts' (Gómez-Pérez, et al 2004). Therefore, concepts represent the abstractions of actual objects and events. But, concepts do not represent the specific details of each and every actual object or event. The instances of concepts, however, capture the specific detail of each actual object or event. For example, the concept *MachineTool* is a term used to represent all the different machine tools. Similarly, concepts *Operator*, *Shop*, and *ProcessPlan* could represent instances like Tom, disc_machining_shop-1, and Standalone_disc_processplan-123 respectively. Thus, concepts form a Meta level structure for the actual shop floor knowledge composed of instances. Typically in a knowledge base system the knowledge composed of instances has an underlying Meta level structure. This Meta level structure is composed of concepts and is used to reason about the knowledge composed of instances.

In this thesis, however, there is a need to have not just a single knowledge structure and the instances in it but, to have multiple levels of knowledge abstraction. This means that the instances of one structure can also be used as a structure for instantiating more specific knowledge. Thus, the Meta level structure which is composed of concepts should also behave like an instance. This leads to a need for defining multiple levels of knowledge abstraction.

Based on the understanding of knowledge levels gained from Turban and Aronson (2005) and Gómez-Pérez et al (2004), two knowledge levels have been defined in the context of this thesis. A brief description of these knowledge levels is given below.

4.5.1.1 Individual Level Knowledge

The term 'Individual' has been used to refer to the instances which cannot be instantiated further (IODE 2010). Therefore, a level of knowledge which cannot have further instances is referred as '*Individual level knowledge*' in this thesis. This knowledge cannot act as a structure for any other knowledge but this knowledge is itself based on an underlying Meta level structure.

For example, consider the sentence 'Tom has roll number R-123'. Neither Tom nor his roll number can have further instances. However, in a knowledge base system both Tom and his roll number will exist as instances of some concepts e.g. '*Student*' and '*Roll Number*'. These concepts may be linked through the relation *hasRoll Number*'. Thus, these concepts and relations provide a structure for instantiating the students, their roll numbers and for relating the roll numbers to the students. But the instantiated knowledge like 'Tom has roll number R-123' cannot be instantiated and this is an example of the *individual level knowledge*.

4.5.1.2 Meta Level Knowledge

In this thesis a concept is considered to be equivalent to a class. Concepts can be instances of other concepts and can also have instances of their own. Therefore, concepts are not individuals. Based on this interpretation of concepts it can be stated that the *Meta level* structure is composed of concepts.

In this thesis, the *Meta level* structure in which the *individual level knowledge* is preserved is also considered a type of knowledge. Thus, a Meta level structure underlying the *individual level knowledge* can be referred to as '*Meta level knowledge*'. An example of this is the structure for instantiating the sentence 'Tom has roll number R-123'. The structure for this sentence is composed of concepts and relation which can be stated as 'Student hasRollNumber Roll Number' where both 'Student' and 'Roll Number' are concepts and 'hasRollNumber' is the relation. This structure is at the Meta level and can be called *Meta level knowledge* in the context of this work.

A requirement to capture and reason about *Meta level knowledge* has been identified in this thesis. This implies that the *Meta level knowledge* has to behave as an instance. But in order to do this, a structure underlying *Meta level knowledge* should be defined. This is required to act as a reference for capturing, querying and

retrieving the *Meta level knowledge*. Thus, a MetaMeta level structure underlying the *Meta level knowledge* is required. Therefore, It is much more difficult to deal with *Meta level knowledge* (Turban and Aronson 2005) and a typical knowledge modelling methodology is not sufficient to meet this requirement. The next section describes the requirements to capture production knowledge at multiple level of knowledge abstraction.

4.5.2 Knowledge Abstraction Levels in Production:

Production engineers often deal with knowledge at different levels of abstraction. Production engineers' aim is to capture the detailed process plans at the *individual level knowledge*. An example of a detailed individual level process plan is shown in figure 4.3. This has detailed specifications of Operations with their order as well as date and time, specifications of Machine Tools, Fixtures and Cutting Tools.

PROCESS PLAN FOR STAND ALONE AERO ENGINE DISC						
ProcessPlan: PPST12345-STAL-Disc-Part No. HPC-T3000-St9-76890						
Sr. No.	Operation		Setup		CuttingTool	Date & Time
			MachineTool	Fixtures		
1	OP-10	Turning Outer Dia	TurningStepOp10		Tool Holder XY 14Z, Insert 123x-3B Grade TG 19 Insert 143y-3A Grade TG 20	10/05/2011 08:30:00 to 1:00
			3-Axis Turning Center Cincenati-Ab1 Axis Range in cm Chuck Size D-50, Z-1200	Clamp Ring X23, Adaptor Plat XU23, Back Plate V12		
2	OP-20	Mill Disc Web	MillingSetupOp20		Ball Mill cutter A50 Grade B2, Ball Mill cutter A30 Grade A1,	10/05/2011 01:30 to 16:30
			3-Axis Machining Center-Ac1 Axis Range in cm X-430, Y-300, Z-400	Sleeve Fixture Z90-Seating Pads X90		
3	OP-30	Drill Bolt Holes	DrillingSetupOp30		Drill Bit X290x-3B Grade DR 67 Length 75mm, Dia 15mm, Flutes 3,	11/05/2011 08:30 to 10:00
			3-Axis Machining Center-Ac1 Axis Range in cm X-430, Y-300, Z-400	Sleeve Fixture Z156, seating Pads 130		
4	OP-40	Mill Loading Slot	MillingSetupOp40		Ball Mill B 67Z-5A Grade 5M Special Insert A-25deg	11/05/2011 10:30 to 15:30
			5-Axis Machining Center MC531	Clamp Ring X27, Calmping Unit Ab19, Back Plate V16		
5	OP-110	Inspection	InspectionSetupOp110		Probes M10, B20, Profile Prob M60	12/05/2011 08:30 to 17:00
			CMM IM-558, X500, Y 500, Z800	Dovel Pins, Seating Pads		

Figure 4.3: An example of detailed individual level process plan constituted of individuals or instances

As mentioned in section 3.2.2, production engineers during the early phases of product development, are often not interested in the details like date and time of operation (e.g. date: 10/05/2011 and time: 8:30-1:00, operation number e.g. OP-30, and detailed machine tool & cutting tool specifications). Production engineers are interested in the abstractions of such details e.g. the kind of operations like milling, drilling, and turning, their sequence, the kind of machine tools like machining centre. Thus, they are interested in a Meta level view of the detailed process plans. An example of such a Meta level process plan is shown in figure 4.4. Such Meta level knowledge is required during early stages of product development to make decisions

about the manufacturability of parts with the available resources. Production engineers may also want to query the manufacturability of certain new features within a previous part family without going into the specific details of Machine Tool, Fixtures, Cutting Tools, Dimensions and Tolerances.

HIGH LEVEL ABSTRACT/META PROCESS PLAN FOR HIGH PRESSURE COMPRESSOR AERO ENGINE DISC				
Sr. No.	Operation	Setup		CuttingTool
		MachineTool	Fixtures	
1	Turning Op	TurningOpSetup		Verticle Turning tool
		Turining Center	Clamp Ring, Adaptor Plat, Back Plate	
2	Rough Milling Op	RoughMillingOpsetup		Ball Mill cutters
		Machining Center	Sleeve Fixture, Seating Pads	
3	Drilling Op	DrillingOpSetup		Drill
		Drilling Machine	Sleeve Fixture, seating Pads	
4	Finish Milling Op	FinishMillingOpsetup		Finish Ball Mill
		5-Axis Machining center	Clamp Ring, Calmping Unit Back	
5	Inspection Op	InspectionOpSetup		Touch Probes, Profile Prob
		Coordinate Measuring Machine	Dovel Pins, Seating Pads	

Figure 4.4: An example of abstracted/ Meta level ProcessPlan consisting of abstract concepts

An abstract process plan will consist of concepts instead of individuals. Figure 4.5 presents abstracted concepts and the way they provide a Meta level structure for

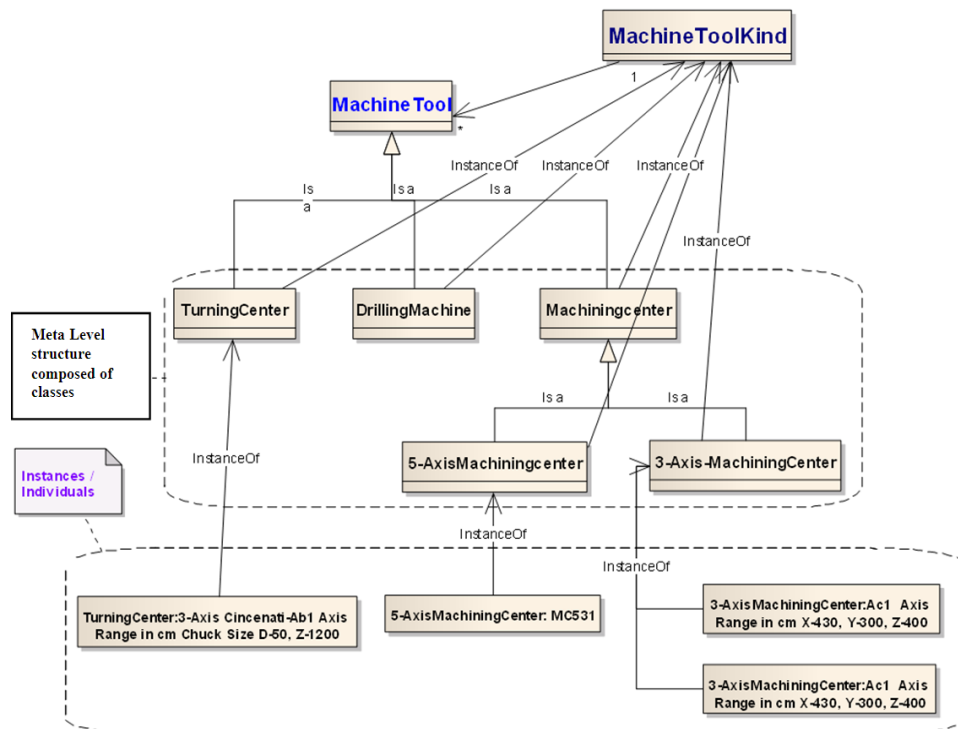


Figure 4.5: Depiction of multiple levels of abstractions through MachineTool concepts and individuals. Concepts *Turning Centre*, *Machining Centre*, and *Drilling Machine* are

abstracted representation of actual machine tools and are therefore grouped under the concept '*MachineTool*'. Because the concepts grouped under the concept *MachineTool* represent different kinds of machine tool, they can, therefore, be considered instances of another concept '*MachineToolKind*'. This is one example and similar conceptualization approach can also be applied to concepts *ProcessPlan*, *CuttingTool*, and *Operations* which can be used in combination to generate an abstract process plan.

A hypothetical example of an abstract *ProcessPlan* built by using abstract concepts is shown in figure 4.4. Note that there are no detailed specifications but only abstractions. This sort of *process plan* can provide a Meta-structure for any number of individual level process plans like the one shown earlier in figure 4.3.

Conceptualization for representing abstractions of individuals have been discussed in the literature (Henderson-Sellers 2011; Henderson-Sellers and Gonzalez-Perez 2005; Henderson-Sellers and Hawryszkiewicz 2008; Palmer et al. 2011). However, the capture and sharing of Meta level manufacturing knowledge remains to be explored. There are two main research questions in this regard.

1. What are the required concepts and relations that can support the capture and reasoning about Meta level manufacturing knowledge?
2. How can the Meta level manufacturing knowledge be formally represented, captured and reasoned about using those *concepts*?

The proposed solution to these questions is presented in chapter 7 of this thesis.

4.6 Summary

This chapter has presented the main argument for the thesis and has highlighted the novel aspects. The main novel aspect regards the development of a heavyweight manufacturing core concepts ontology which, supports the development of application specific product design and production ontologies and also facilitates knowledge sharing across them. Another aspect was regarding the need to capture the varying depths of meaning of manufacturing concepts and a further aspect was to identify a method to capture and reason about the *Meta level knowledge*. The chapter has presented the justification and need to be working on these aspects of research. The development of these aspects and their experimental validation is presented in the following chapters.

5 Manufacturing Core Concepts & Relations and their Formalisation

5.1 Introduction

This chapter is developed against the issues raised in section 4.3 of chapter 4 about the need to identify a set of core manufacturing concepts and relations and the formal capture of their semantics to assist knowledge sharing across product design and production. The identification of concepts and relations, developing their hierarchies and formally defining them is in line with point 3 of the ontology development methodology outlined in section 1.4.2.

This chapter presents the development of the manufacturing core concepts ontology (MCCO) by identifying a set of core manufacturing concepts and relations from several sources and formalising them in heavyweight logic. The concepts in MCCO should provide a base for developing product design and production ontologies and supporting interoperability across them. It is claimed that the MCCO is equipped with the ability to understand the semantics of concepts. It is also claimed that the MCCO supports the development of specialized product design and production ontologies and also provides a route to knowledge sharing across these domains. These claims are experimentally investigated in chapter 8.

The chapter is organised following the step 3 (ontology building) of the adopted ontology development methodology (section 1.4.2). The first three points in building the ontology are to identify the concepts and relations, build hierarchies and add relations to the hierarchies. Therefore, section 5.2 reports the exploration of the core manufacturing concepts by dividing them into several categories. This section also reports the intra-category relations (between the concepts belonging to each category) and presents the hierarchies of concepts for each category. Section 5.3 discusses the inter-category (between the concepts belonging to different categories) relations and an overall hierarchical model of the ontology is presented in section 5.4. According to the final two points of the ontology building step, section 5.5 describes the understanding of the constraints required to capture the semantics and the capture of those semantics in formal logic through examples. Section 5.6 presents a summary of the chapter.

5.2 Exploration of Core Concepts and Intra-Category Relations

Manufacturing knowledge has contributions from several different sources including manufacturing resources, manufacturing facilities, manufacturing process, and historical knowledge about manufactured parts. An exploration of concepts to capture knowledge from these different sources has helped to identify a number of relevant models, ontologies and standards that acknowledge important manufacturing concepts. The more relevant ones have been mentioned in table 2.5. The industrial study also provided an understanding of the core manufacturing concepts.

The first step in developing a manufacturing ontology as mentioned in section 4.3 of chapter 4, is to synthesize the understanding gained from the different models, ontologies, standards and industrial study into a set of core manufacturing concepts and relations which can effectively support the knowledge sharing across the product design and production domains. Based on the analysis of the problem domain from several different sources (tables 2.5 and chapter 3) it has been concluded that, to meet the needs of the proposed ontology, eight different sets of information need to be explored. In order to do that, the manufacturing concepts from the reviewed models, ontologies, standards and the industrial study have been categorized into eight main concepts as shown in figure 5.1. These are named as *RealisedPart*, *PartVersion*, *ManufacturingFacility*, *ManufacturingResource*, *ManufacturingMethod*,

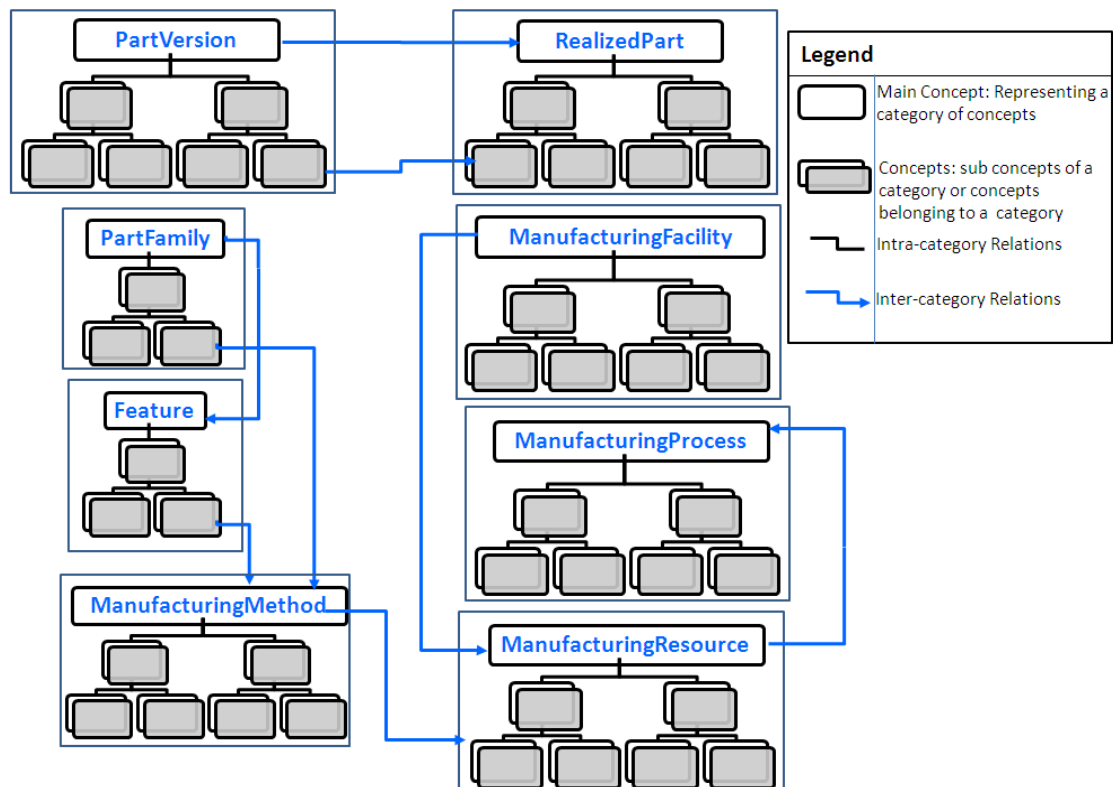


Figure 5.1: Categories of manufacturing core concepts and relations

ManufacturingProcess, *Feature*, and *PartFamily*. Each of these concepts represents a category of concepts. Each category is explored for several concepts which can be included in that category.

The first activity in ontology development has been to elicit key terms in an attempt to create the backbone of concepts for each category in the MCCO. Furthermore, relations have also been defined. This is because terms in an ontology ought to have relations with each other to support the development of some understanding about the concepts. Relations are required to formally capture the meanings of the terms. Relations also help to describe the context of the concepts and help in defining the behaviour of concepts. Therefore, the relations between the eight main concepts and between the concepts belonging to each main category are explored.

The relations have been divided into '*Intra-category relations*' and '*Inter-category relations*'. The *intra-category relations* describe the relations existing within the concepts belonging to each main category. The concepts which link concepts from one category to the concepts from another category are termed as *inter-category relations* in this thesis. The *inter-category relations* are discussed in section 5.3 of this chapter. On the other hand, the *intra-category relations* are discussed in sections 5.2.1 to 5.2.8 along with the exploration of concepts belonging to each category. This helps in better understanding and defining the concepts belonging to each category.

5.2.1 Realised Part

5.2.1.1 Realised Part concepts

Production is concerned with producing products or parts. A *Part* is a “discrete object that can come into existence as a consequence of a manufacturing process (ISO/TS-10303-1022 2004)”. In the process of realisation, *Parts* may have a number of different states. For example a realised part can be a prototype, a rejected *Part* or work in progress. Each state has important knowledge associated to it and that knowledge needs to be captured. The capture of that knowledge requires formalisation of different states of the part. Examples of these states are part in specification, prototype, rejected part and work in progress as shown in figure 5.2.

The root concept that needs to be defined in this category is *RealisedPart*. This concept as shown in figure 5.3 has appropriately been transitively subsumed under the concept *Object* and *Part*. The concept *Object* is taken from the foundation ontology ULO. The use of foundational concepts for defining core concepts provides consistency in semantics (Borgo and Leitão, 2007)

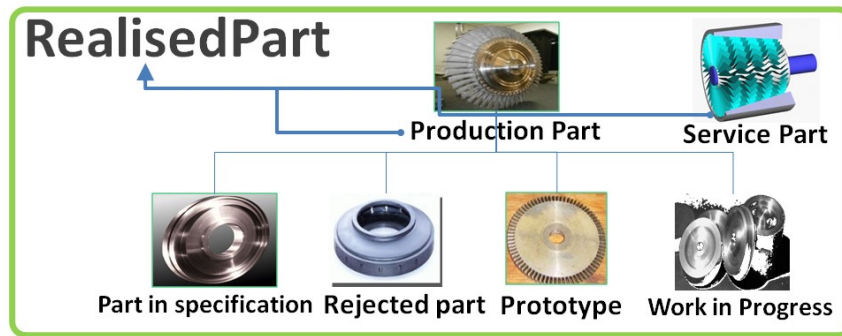


Figure 5.3: Examples of realised parts

The subsumption of *RealisedPart* under the concept *Part* is obvious because a *RealisedPart* is one of the conceptualizations that a *Part* can have. This is evident from the concept '*DesignedPart*' shown in figure 5.3. *DesignedPart* represents a virtual *Part* that has not necessarily been realised. The conceptualization provided in (ISO/TS-10303-1164 2004) of a *RealisedProduct* has been adapted to define the concept *RealisedPart*. The informal definition of *RealisedPart* following the standard definition (ISO/TS-10303-1164 2004) is "The concept *RealisedPart* represents a part that exists physically in the real world and whose properties can only be known by observation". It is to be noted that the concept *Part* has been used here as a substitution for the concept *artifact* in the ISO definition.

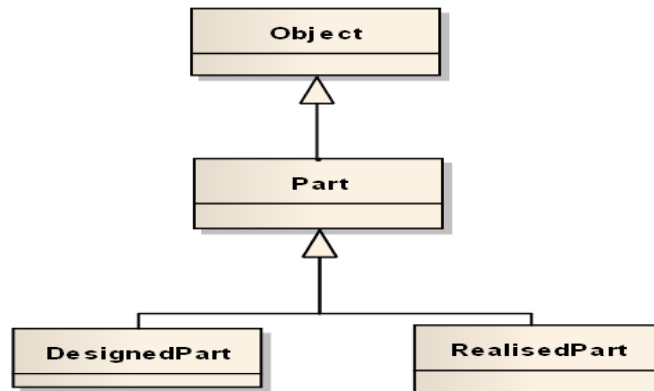


Figure 5.2: Subsumption of RealisedPart under concept Object

A *RealisedPart* can be a manufactured part or a service part. The concepts identified to capture these conceptualizations are *ProductionPart* and *ServicePart* as shown in figure 5.4. Since this work is not exploring the service domain, the conceptualization of *ServicePart* is not explored further but the conceptualization of *ProductionPart* is. The definition of the concept *ProductionPart* is adapted appropriately from the MANDATE standard (ISO-15531-1 2004) and (ISO/TS-15926-4 2007) as "The

concept *ProductionPart* refers to a *RealisedPart* produced through a production method”

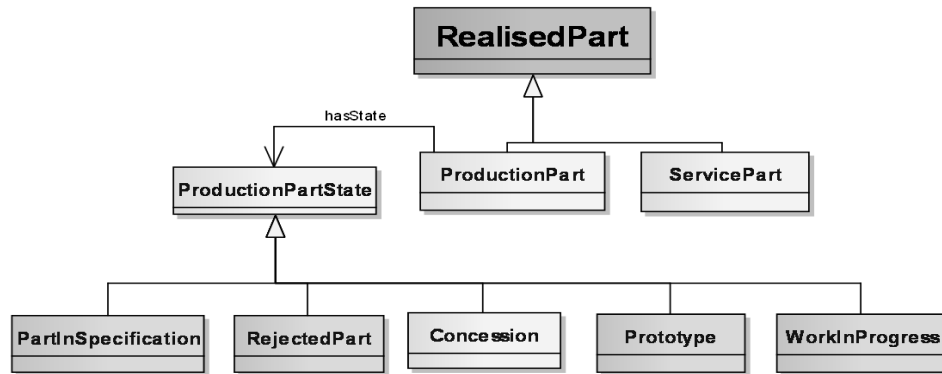


Figure 5.4: Lightweight representation of RealisedParts concepts

The different states of a *ProductionPart* are represented by the concepts, *PartInSpecification*, *RejectedPart*, *Concession*, *Prototype* and *WorkInProgress*. A concept '*ProductionPartState*' is introduced to refer to the states of *ProductionPart*. The concepts representing different states of the *ProductionPart* are subsumed under the concept '*ProductionPartState*' as shown in figure 5.4 and the informal definitions of these concepts are as follows;

“The concept *PartInSpecification* represents the state of a *ProductionPart* which meets the design specifications”

“The concept *RejectedPart* represents the state of a *ProductionPart* that does not meet the design specifications and cannot be reworked or corrected”

“The concept *Concession* represents the state of a *ProductionPart* that is slightly out of the design specifications but which can be considered for acceptance by customers”.

“The concept *Prototype* represents the state of a *ProductionPart* which represents the initial physical trial model of a *DesignedPart*”

“The concept *WorkInProgress* represents a state of a *ProductionPart* which is yet to go through some of the production operations”.

5.2.1.2 Realised Part's Intra-Category Relations

These different concepts are not subsumed under the concept *ProductionPart* as mentioned earlier but rather under the concept '*ProductionPartState*'. This is because a *ProductionPart* can have multiple states simultaneously. For example, a *ProductionPart* can have the states of *Prototype* and *PartInSpecification* at the same

time. Therefore, the core concepts representing the states of a *ProductionPart* are related to the *ProductionPart* by the relation ‘*hasState*’. This enables the system to simultaneously capture the multiple states of a *ProductionPart*. This relation is shown in figure 5.4 and can be stated as “*ProductionPart hasState ProductionPartState*”. This relation applies between the concept *ProductionPart* and all the concepts subsumed under the concept *ProductionPartState*.

5.2.2 Part Version

During the industrial exploration it was observed that each part had a number of revisions in its design and production plan. Important design and production knowledge was associated with each version of the *Part*. In this category, the concepts and relations that facilitate the capture of this knowledge are explored.

5.2.2.1 Part Version Concepts

The lightweight formalisation of the concepts and relations identified in this category are shown in figure 5.5. The first concept that needs to be defined is *PartVersion*. *PartVersion* is conceptualized based on definition given in ISO/TS-10303-1022 (2004) as “The concept *PartVersion* represents a version of a Part which captures the history of the Part”

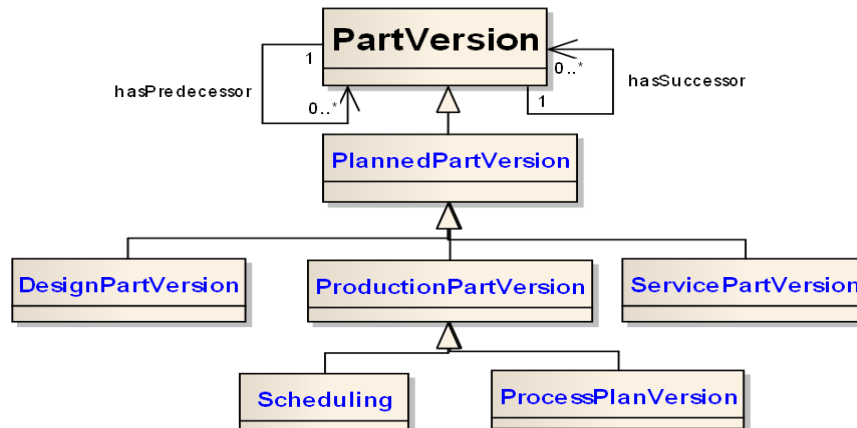


Figure 5.5: Lightweight formalisation of PartVersion concepts and relations

An important set of information in the *PartVersion* from the planning perspective is a ‘*PlannedPartVersion*’. A *PlannedPartVersion* is defined based on the standard definition (ISO/TS-10303-1164 2004) as “The concept *PlannedPartVersion* represents historical versions of parts planned for realisation”

A *PlannedPartVersion* represents history of the planned parts from design, production and other perspectives. Therefore, *PlannedPartVersion* has been subcategorized into *DesignPartVersion*, *ProductionPartVersion*, and *ServicePartVersion* as shown in

figure 5.5. The definition of *ProductionPartVersion* can be stated as “The concept *ProductionPartVersion* represent the history of planned parts for production” The definition of *DesignPartVersion* is similar whilst definitions for *PartVersion* regarding other domains are beyond the scope of this work.

5.2.2.2 Part Version's Intra-Category Relations

A *PartVersion* captures the history of some *Part*, and therefore the relation across different versions with respect to their preceding and succeeding *PartVersion*(s) needs to be captured. This is done through the relations *hasPredecessor* and *hasSuccessor* as shown in figure 5.5.

Other intra category relations are as follows:

- *PartVersion* subsumes *PlannedPartVersion*
- *PlannedPartVersion* subsumes *ProductionPartVersion*, *DesignPartVersion*, and *ServicePartVersion*

5.2.3 Manufacturing Resource

5.2.3.1 Manufacturing Resource concepts

Figure 5.6 shows different examples of manufacturing resources for which the concepts are explored in this section. In accordance with the scope of this thesis, *ManufacturingResources* are key to the capture of production knowledge. The reviewed literature has helped in defining the concept *ManufacturingResource* as explained in the following discussion.

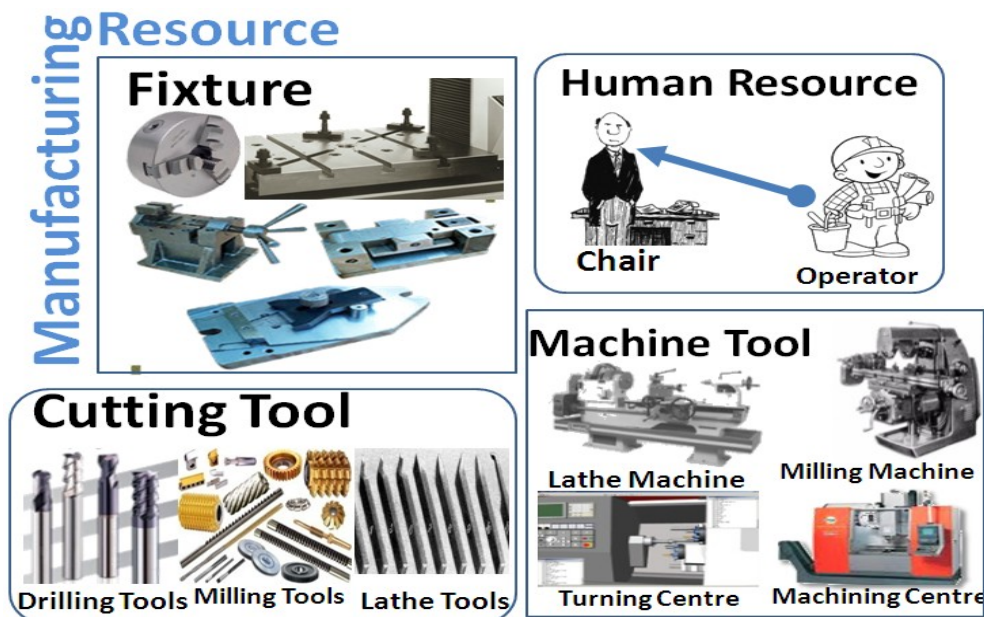


Figure 5.6: Examples of ManufacturingResource concepts

ManufacturingResources have been categorized in the MANufacturing Semantics ONtology (MASON) by Leimagnan et al (Leimagnan et al. 2006) as shown in figure 5.7. Some concepts and their categorization in the MCCO are influenced by MASON but with some required variations. For instance, in MASON concepts equivalent to *RealisedPart* and *ManufacturingFacility* concepts are kept under resources, whereas, in the MCCO *RealisedPart* and *ManufacturingFacility* represent two of the eight main concepts and they are not subsumed under *ManufacturingResource* concepts.

Semere's work helped in developing an understanding of the *ManufacturingResource* concepts. Semere et al (2007) had the concept '*MachiningResource*' as one of the main concepts in their machining ontology. They subsumed *MachineTool*, *CuttingTool*, and *MachiningAttachments* under the concept *MachiningResources* where each one of the subsumed concepts itself subsumed other concepts.

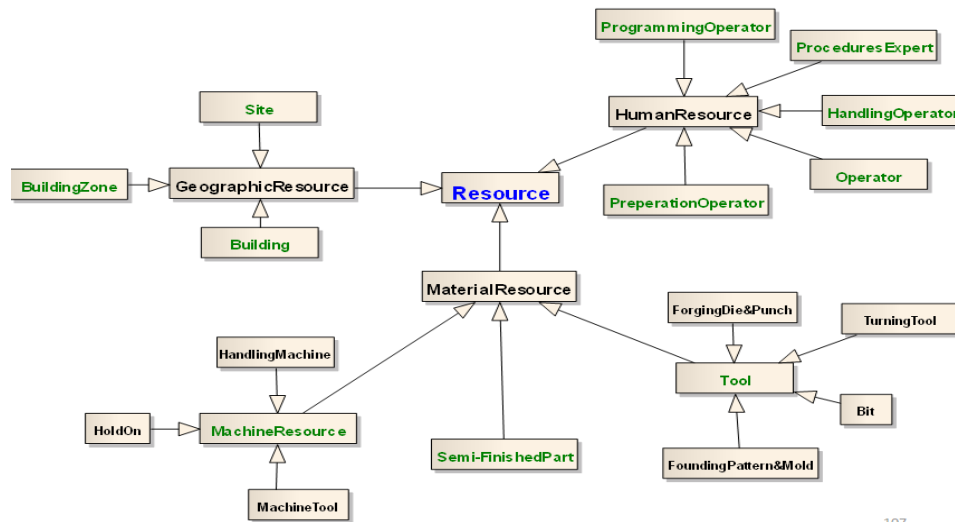


Figure 5.7: Manufacturing Resources hierarchy adapted from (Leimagnan et al. 2006)
Based on the above mentioned works, a lightweight UML model of *ManufacturingResource* concepts is developed as shown in figure 5.8. Keeping in view the requirements of this thesis and the discussion above, the concept *ManufacturingResource* is defined as “The concept *ManufacturingResource*

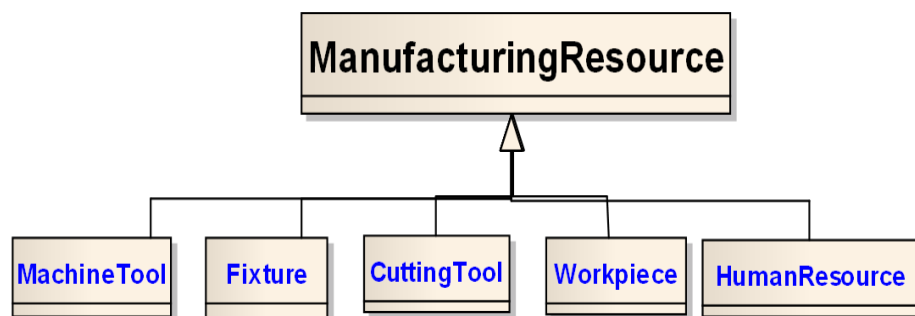


Figure 5.8: Lightweight representation of ManufacturingResource concepts

represents the resources required for the production of *Parts* and *Features*”

Some of the *ManufacturingResource* concepts are generic to the whole of the product lifecycle like *HumanResource*, but others are more focused on production e.g. *MachineTool*, *Fixture*, and *CuttingTool*. The concept *WorkPiece* is classed as a *ManufacturingResource* and is therefore, subsumed under *ManufacturingResource*. This concept is defined as “A *ManufacturingResource* on which manufacturing processes are performed directly.”

It is important to distinguish between the concepts *WorkPiece* and *WorkInProgress* (from *RealisedPart* category). *WorkPiece* represents an input to the production method and is therefore categorized as *ManufacturingResource*, whereas, the concept *WorkInProgress* represents a *Part* that has been worked on but still has to go through some operation(s). These two concepts may refer to the same part during certain manufacturing stages but their semantics and the information captured by them will remain different. The input materials e.g. billet, slab, bar, and rod are also examples of *WorkPiece*.

The concept *CuttingTool* is defined based on ISO-10303-224 (2006) as;

“The concept *CuttingTool* represents a *ManufacturingResource* used to directly remove material from a *WorkPiece*”

Another important *ManufacturingResource* concept is *Fixture* which is defined based on the ISO-10303-224 (2006) definition of *Fixture* as;

“The concept *Fixture* represents a *ManufacturingResource* used for holding and locating the *WorkPiece* or *CuttingTool* in position”

The *HumanResource* concept from MASON is included in the ontology. In the MASON ontology the definition of *HumanResource* has not been given explicitly. Therefore, a textual definition of *HumanResource* is presented as;

“The concept *HumanResource* represents the *ManufacturingResource* which plans and handles all other *ManufacturingResource(s)*”

MachineTool is an important concept in the manufacturing core concepts ontology. Semere et al (2007) refer to this concept as “*MachineTool* can use *CuttingTool* and other machine attachments”. *MachineTool* is also classified as a

ManufacturingResource in ISO standard (ISO-16100-1 2009). These works helped in defining *MachineTool* as;

“The concept *MachineTool* represents the *ManufacturingResource* on which *Fixture(s)*, *CuttingTool* and *WorkPiece* are setup”

The above mentioned concepts in this section are core *ManufacturingResource* concepts. Although further sub-concepts of these concepts can be created, this is not done in the MCCO, because it is necessary to maintain the core level nature of the ontology. Further classifications using these core concepts are left to the application domain ontologists.

However, the detailed hierarchies of each of these core concepts have been further developed as extensions of the core concepts. The commitment to the extension of MCCO remains optional for the domain ontologists. For instance; *Fixture* can be classified with respect to the *WorkPiece* holding and *CuttingTool* holding. *HumanResource* can subsume concepts like *Engineers*, *Designers*, *Manufacturing Planner*, and *Operator*. The concept *Operator* can subsume concepts like *HandlingOperator* and *Preparatory Operator*. A hierarchy of *CuttingTool* concepts can be made based on the *ManufacturingProcess* they perform. *MachineTool* can be classified based on the classification in machining ontology by Semere et al, (2007). The lightweight as well as heavyweight formalisation of these hierarchies is presented as extensions to the MCCO in section A2.6 of appendix A2.

5.2.3.2 Manufacturing Resource's Intra-category Relations

The *intra-category* relations for *ManufacturingResource* are simple subsumption relations. The concepts *MachineTool*, *Fixture*, *CuttingTool*, *WorkPiece*, and *HumanResource* are all subsumed under *ManufacturingResource*.

5.2.4 Manufacturing Facility

5.2.4.1 Manufacturing Facility Concepts

The exploration of *ManufacturingFacility* concepts starts from the work done on the facilities model. The first facilities model was developed by (Simpson et al. 1982) and later used and extended by Molina et al (1995) and Zhao et al. (1999). Zhao et al (1999) proposed that a *ManufacturingFacility* at its lowest level can be considered to be an individual work station like a machine, assembly station or a design bench and at higher levels a *ManufacturingFacility* can be a manufacturing cell, shop, factory or an enterprise. Hence, Zhao's model has been utilised with addition of the concept '*Location*' as shown in figure 5.9. The concept *Location* is added to identify the

location of the facility. The informal definitions of the concepts shown in figure 5.9 are as follows;

“The concept *ManufacturingFacility* represents the object which houses the

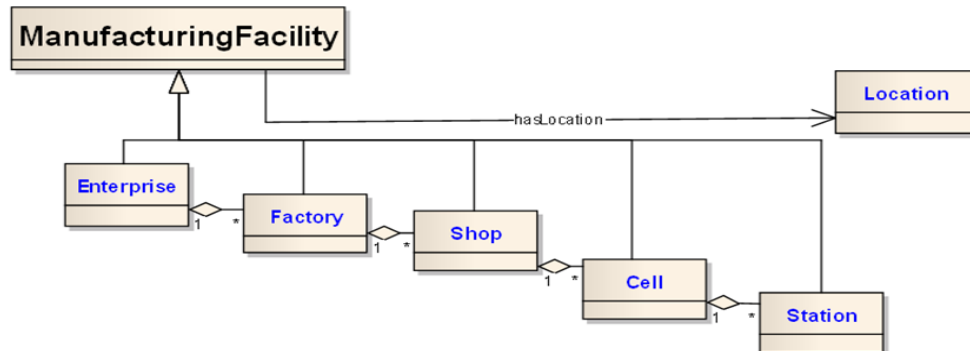


Figure 5.9: Manufacturing Facility lightweight representation adapted from (Zhao et al, 1999) *ManufacturingResource*”.

Other concepts subsumed under the *ManufacturingFacility* are defined as follows;

“The concept *Station* represents a *ManufacturingFacility* consisting of a single working station.”

“The concept *Cell* represents a *ManufacturingFacility* consisting of multiple *Stations* grouped to perform similar tasks.”

“The concept *Shop* represents a *ManufacturingFacility* consisting of multiple *Cells* grouped to manufacture *Parts* that are similar from a production perspective.”

“The concept *Factory* represents a *ManufacturingFacility* consisting of multiple *Shops* to produce a single *Part*, *Product*, set of *Part*, set of *Products*, or services”

“The concept *Enterprise* represents a *ManufacturingFacility* consisting of multiple *Factories* grouped to contribute towards a common *Product*, set of *Products*, or services.”

5.2.4.2 Manufacturing Facility’s Intra-Category Relations

The *intra-category relations* for *ManufacturingFacility* concepts are mainly aggregation relations where one facility has an aggregation relation with the other. For example, *Enterprise* has an aggregation relation with *Factory*, *Factory* has an aggregation relation with *Shop*, and so on. This means that an *Enterprise* can have several *Factory(s)*, a *Factory* can have several *Shops*, a *Shop* can have several *Cell(s)*, and a *Cell* can have several *Station(s)* as depicted by the one to many (1..*) relations in figure 5.9.

Every *ManufacturingFacility* has a location e.g. a disc production factory in Derby or a disc production factory in Sunderland. Therefore, a relation '*hasLocation*' has been defined to capture the location of *ManufacturingFacility* as shown in figure 5.9.

5.2.5 Manufacturing Method

The concept '*ManufacturingMethod*' is defined as a generic term applicable to different product lifecycle disciplines. The informal definition is as follows;

"The concept *ManufacturingMethod* represents a sequence of events involved in the manufacture of a *Part*"

According to the above definition, *ManufacturingMethod* can represent the service method, maintenance method or production method. In the context of this thesis, the focus is on the capture and reasoning about production methods for *Feature(s)* and *PartFamily(s)* at multiple levels of knowledge abstraction (section 4.5.2 of chapter 4). The approach to capture the *ProductionMethod* at *Meta Level knowledge* is complex and is explained in detail in chapter 7. This section only explains the concepts, relations and structure for the capture of the *ProductionMethod* at the individual level

5.2.5.1 Production Method Concepts

As a first step towards identifying concepts for the capture of *ProductionMethod*, the identified core concepts *PartFamily*, *Feature*, *MachineTool*, *Fixture*, and *CuttingTool* are also utilized. The first concept '*ProductionMethod*' is informally defined as "a sequence of events which describe the procedure for the production of a *Part*, *PartFamily* or *Feature*." The concept representing the structure for an *individual* level *ProductionMethod* is termed as a '*ProcessPlan*' for parts and '*FeatureProductionMethod*' for features in the thesis.

One of the most relevant works in developing and understanding of the *ProcessPlan* and *FeatureProductionMethod* concepts is perhaps by Gunendran and Young (2010) who presented a model to capture the best practice *Feature* and *PartFamily* knowledge. The terms having similar conceptualisation as those of *ProcessPlan* and *FeatureProductionMethod* are 'M/C Sequence' and 'Stage Sequence' respectively in Gunendran and Young's model (2010). In Gunendran and Young's model a *ProcessPlan* is inferred from the sequence of machines. In this thesis however, a *ProcessPlan* is defined as "a sequence of *operations*". This is because in a *ProcessPlan* a single machine can perform multiple different operations. Moreover, it is also established to define *ProcessPlan* as a sequence of operations (Borgo and Leitão 2007; ISO-10303-49 1998).

The term *Operation* is conceptualised informally as “an event in the *ProcessPlan* that has a unique *Setup*” based on the understanding from Borgo and Leitão (2007), standard (ISO-15531-43 2006) and Gunendran and Young (2010).

Similar to a *ProcessPlan* the *FeatureProductionMethod* has been informally defined as “a sequence of *Stages*” based on Gunendran and Young’s (2010) model where *Stage* is “an event in the *FeatureProductionMethod*, which is performed with a unique setup”. The lightweight formalisation of the concepts discussed so far in this section is represented by the highlighted portion of figure 5.10.

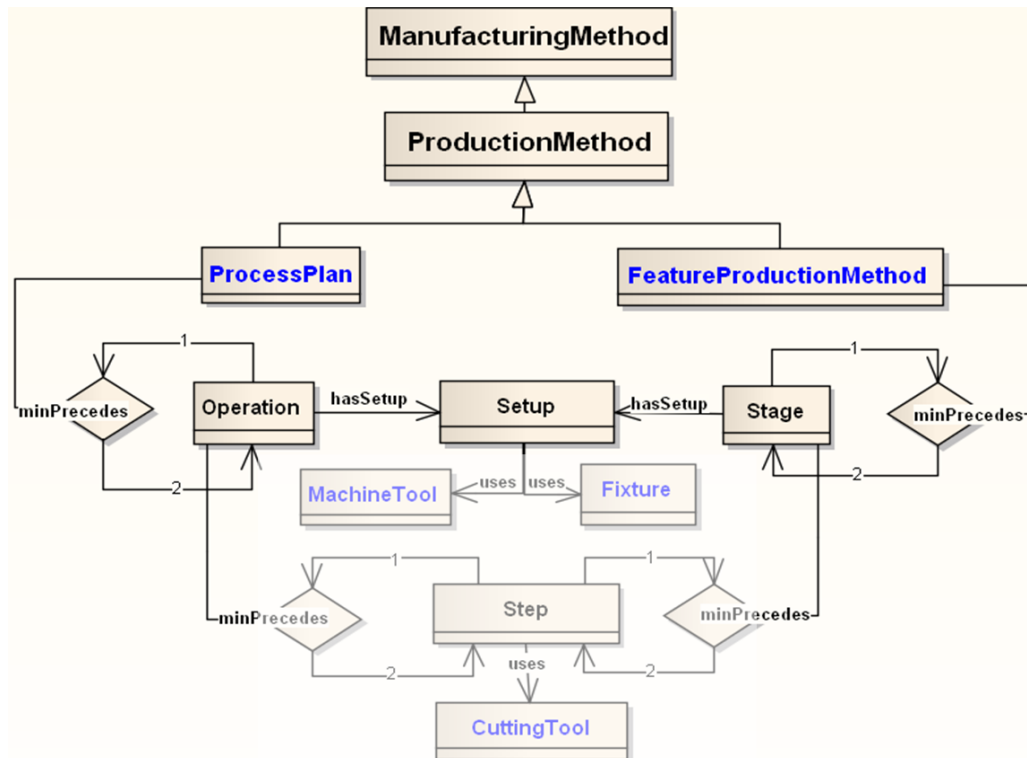


Figure 5.10: Lightweight formalisation of *ProcessPlan* and *FeatureProductionMethod*

Figure 5.10 shows the lightweight weight formalisation of *ProcessPlan* on the left and of *FeatureProductionMethod* on the right. In figure 5.10, the relation *minPrecedes* represented by the diamond shape is a ternary relation that has been taken from the PSL (ISO-18629) to capture the sequencing of events in a production method. For example, in a *ProcessPlan* one *Operation* precedes (*minPrecedes*) another *Operation*. The relation *minPrecedes* is explained in detail in section 7.3.2 of chapter 7.

The conceptualization of *Operation* and *Stage* in this work is more appropriate than in Gunendran and Young’s (2010) model. This is because they inappropriately differentiated one *Operation* from the others with respect to the unique *MachinTool(s)* that are used for performing *Operation(s)*. Similarly, they inappropriately differentiated

one *Stage* from other *Stage(s)* with respect to the unique *MachinTool(s)* that are used for performing *Stage(s)*. However, a single *MachineTool* cannot be unique to a single *Operation* or *Stage* and can perform multiple different *Operations* and *Stages* within a sequence. For example, the same milling machine can perform different drilling and milling operations in a single process plan with different setups. Therefore, appropriately, one *Operation* is differentiated from others and of one *Stage* from others with respect to their unique setups because they have unique setups. *Setup* is defined as “an event that prepares a *MachineTool* and *Fixture* for performing a unique *Operation* or *Stage*” as shown in figure 5.10.

It is also understood from Gunendran and Young (2010) that each *Operation* and *Stage* consists of a sequence of *Steps* where *Step* is “an event in an *Operation* or a *Stage* that is performed with a specific *CuttingTool*”. The lightweight formalisation of this is shown in the non highlighted portion of figure 5.10.

5.2.5.2 Production Method's Intra-Category Relations

The relation *minPrecedes* is used to capture sequencing of *Operations*, *Stages* and *Steps* as show in figure 5.10. The relation linking the concepts *Operation* and *Setup* is ‘*hasSetup*’. *Setup* itself is related to the concepts *MachineTool* and *Fixture* through the relation ‘*uses*’ which provides the lightweight semantics of *Setup*. The lightweight formalisation of *Step* is provided by relating it to the concept *CuttingTool* through the relation ‘*uses*’.

The relation between *FeatureProductionMethod* and *ProcessPlan* can be established through *Setup* as shown in figure 5.10. This is also an extension to the Gunendran and Young’s model because they related the *FeatureProductionMethod* and *ProcessPlan* through *Steps*. A *Setup* provides the basis to reason about the manufacturability of *FeatureProductionMethods* in *ProcessPlans*.

5.2.6 Manufacturing Process

There is a huge variety of manufacturing processes in practice today. In the core concepts ontology the only core concept included is the ‘*ManufacturingProcess*’. This is done to preserve the core nature of MCCO. The concept ‘*ManufacturingProcess*’ is defined based on the definition in MANDATE (ISO-15531-1 2004) as follows:

“The concept *ManufacturingProcess* represents the processes performed on a raw material to convert it to a finished or semi-finished Part or Product.”

Based on the classification of *ManufacturingProcess* by Todd (1993) and Fend and Song (2003), a set of *ManufacturingProcess* concepts have been defined as an

extension to the MCCO and they are presented in section A2.6.2 of appendix A2. The use of and commitment to *ManufacturingProcess* extension is optional, however, a commitment to the MCCO is required to support knowledge sharing across product design and production.

This research work explores the *ProductionMethod* and processes in detail for *Feature(s)* and *PartFamily(s)*. Therefore, the concepts *Feature* and *PartFamily* need to be defined for defining *ProductionMethod*. The next section presents the exploration of *Feature* and *PartFamily* concepts.

5.2.7 Feature

Feature concepts and relations are key for relating product design and production domains. These should be explored for providing a better and effective route for knowledge sharing between product design and production domains to assist decision making (Chungoora 2010; Chungoora et al. 2010; Gunendran and Young 2010; Gunendran and Young 2008).

The most developed set of concepts and relations in the MCCO are regarding Features. Feature concepts have been fully formalised to explain the suitability of MCCO as a reference manufacturing ontology. The formalisation of *Feature* concepts and how they are used to explain the novel aspects of research and the effectiveness of the MCCO is detailed in chapter 6.

5.2.8 Part Family

5.2.8.1 Part Family Concepts

Like *Features*, *PartFamily* concepts and relations are also of key importance for relating product design and production domains. *PartFamily* concepts and relations have been explored for providing an effective route for sharing production knowledge with product design (Gunendran & Young 2010, Chungoora 2010).

A great deal of understanding of *PartFamily* concepts was gained from the industrial study, ISO standards, and Gunendran & Young (2010). *PartFamily* concepts help to group *Parts* with similar *ProductionMethod* and *Function* for production and design domains respectively. Based on the understanding gained in section 3.2.2.3 and from Gunendran and Young's (2010) work, the concept *PartFamily* is defined as follows;

"The concept *PartFamily* represents a parametric *Part* which represents a *Family* of *Parts* with similar associated information."

The term '*Family*' used in the above definition has been defined before defining *PartFamily* as "a *Group* identified based on a common criteria". In this definition, the term *Group* is a foundation concept taken from the ULO. The concept *Family* is generic to any domain and *PartFamily* is generic to various manufacturing disciplines i.e. design, production, operation and disposal.

In the context of this thesis, *Parts* can be grouped into *Families* with respect to the similarities in their *ProductionMethods* in production domain and with respect to the similarities in functional requirements in design domain. This conceptualization is captured in the concepts '*ProductionPartFamily*' and '*DesignPartFamily*'. The informal definitions of these concepts based on the understanding gained mainly from industrial study and partially from Gunendran and Young (2010) are as follows;

"The concept *DesignPartFamily* represents a *PartFamily* with the same or similar functional requirements."

"The concept *ProductionPartFamily* represents a *PartFamily* with the same or similar *ProductionMethods*."

Figure 5.11 shows the lightweight representation of *PartFamily* concepts and intra-category relations.

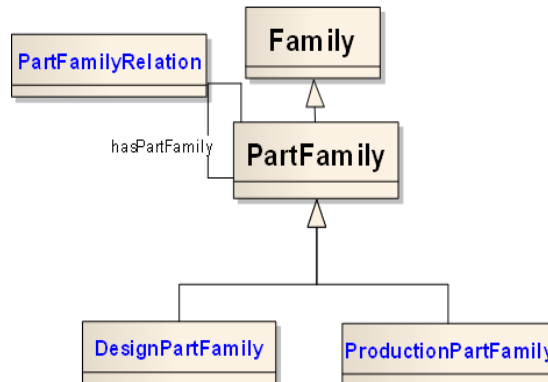


Figure 5.11: PartFamily concepts

5.2.8.2 Part Family's Intra-Category Relations

The intra-category relations between various *PartFamily* concepts are mainly parent-child relations. For example, a *PartFamily* is subsumed under the concept *Family*. Similarly the concepts *ProductionPartFamily* and *DesignPartFamily* are subsumed under the concept *PartFamily*.

There can be relations between different *PartFamilies* belonging either to the same or different domains. In order to capture this association, a relation '*hasPartFamily*' is defined. The relation *hasPartFamily* applies to both *DesignPartFamily* and

ProductionPartFamily because these are subsumed under the concept *PartFamily*. This relation is shown in figure 5.11.

5.3 Inter-Category Relations

This section explores the inter-category relations i.e. the relations between the key categories of information as shown in figure 5.12. These relations form a part of the overall process of formalising the concepts and their semantics. The definitions of many of the concepts in the MCCO are not complete unless their relations with concepts from other categories are defined. For example, the concept *Family* involves the concept *Criteria* in its definition. Therefore, in order to formalise the definition in formal logic, the relation that can relate *Family* concept to *Criteria* is required. Figure 5.12 summarizes the MCCO, its concepts, their inter-category and intra-category relations. The bold lines in figure 5.12 show the *inter-category relations*. The inter-category relations of each category of concepts with the other category are described in the sections 5.3.1 to 5.3.7. Indirectly all the concepts in MCCO are related to each other, therefore, only the direct relations which are necessary to formalise the concepts and capture and share knowledge are defined.

5.3.1 Realised Part's Inter-category Relations

5.3.1.1 With Part Version Concepts

With the production of each new *Part*, there is the potential to gain new knowledge that can be used to improve future process plans. Therefore, the relation *RealisedPart* updates *PlannedPartVersion* has been captured as shown in figure 5.12. Thus *RealisedPart* can update historical knowledge about parts that can be used for future referencing.

5.3.1.2 With Manufacturing Method Concepts

Each *RealisedPart* is based on a *RealisedProcessPlan*. And a *RealisedProcessPlan* is instantiated from a *PartFamilyProductionMethod* as shown in figure 5.12. Therefore, a relation '*isBasedOn*' has been defined to capture the association of a *RealisedPart* with the *RealisedProcessPlan*. This relation is shown in figure 5.12.

5.3.1.3 With Manufacturing Facility Concepts

The *RealisedParts* are produced on a *Station*. That *Station* may be independent or may be a part of the *Cell*, *Shop*, and *Factory*. This means that *RealisedParts* are produced in the *ManufacturingFacility*. A relation '*produces*' has been defined to relate *ManufacturingFacility* to the *RealisedPart* being produced in that facility as shown in figure 5.12.

5.3.2 Part Version's Inter-category Relations with Other Categories

5.3.2.1 With Manufacturing Method Concepts

The *ProcessPlanVersion* (a concept from the *PartVersion* category) supports the development of new *PartFamilyProductionMethods* which helps constitute the *PartFamilyProductionMethods* for new *Parts*. A relation 'supports' has been defined to capture this. This relation is shown in figure 5.12.

It is also understood that the *ProcessPlanVersions* are actually based on the *PartFamilyProductionMethods*. Moreover, every *RealisedProcessPlan* updates the information captured in a *ProcessPlanVersion*. The following relations have been defined to capture these associations.

- *ProcessPlanVersion* 'providesBasisFor' *PartFamilyProductionMethod*.
- *RealisedProcessPlan* 'updates' *ProcessPlanVersion*.

5.3.3 Manufacturing Facility's Inter-category Relations

5.3.3.1 With Manufacturing Process Concepts

The *ManufacturingProcesses* like so many other concepts vary with respect to the *ManufacturingFacility*. Production engineers are interested to have the knowledge about *ManufacturingProcesses* which can be performed in a facility. Therefore, a relation has been defined to facilitate the capture of this knowledge. The relation is termed 'hasCapabilityFor'. This relation is shown in figure 5.12.

5.3.3.2 With Manufacturing Method Concepts

As mentioned earlier in section 5.2.5 the main concept of interest in *ManufacturingMethod* category is *ProductionMethod*. Like many other concepts the *ProductionMethod* is also dependant on the *ManufacturingFacility*. The *ProductionMethod* is updated by the *ManufacturingFacility*. Therefore a relation 'updates' has been defined to capture this association. This relation is represented in figure 5.12.

5.3.4 Manufacturing Resource's Inter-Category Relations

The *ManufacturingResource* concepts are indirectly related to the whole of the MCCO. However, there is only one very important inter-category relation of *ManufacturingResource*. This relation is with *ProductionMethod*.

5.3.4.1 With Manufacturing Method Concepts

ManufacturingResources are utilized in *ManufacturingMethod*. The most relevant of *ManufacturingMethod* concept in this thesis is *ProductionMethod*. The *ProductionMethods* use *ManufacturingResource* like *MachineTool*, *CuttingTool* and

Fixture. Therefore, relations have been defined to capture these associations. As mentioned in section 5.2.5.2, the concepts *Setup* (from *ProductionMethod* category) is related to the concepts *MachineTool* and *Fixture* (from *ManufacturingResource* category) through the relation ‘uses’. In a similar way the concept *Step* is related to the concept *CuttingTool* through the same relation ‘uses’.

Because the relation ‘uses’ relates several *ManufacturingResource* concepts to the *ProductionMethod* concepts, therefore, a relation ‘uses’ has been defined between the main concepts *ManufacturingResource* and *ProductionMethod* as shown in figure 5.12. This makes ‘uses’ applicable to all the concepts belonging to the two categories.

5.3.5 Feature’s Inter-category Relations

The inter-category as well as intra-category relations of *Feature* concepts are explained in detail in chapter 6. However, a brief overview of the inter-category relations of *Feature* concepts is given in this section. There is one key relation which relates *Feature* concepts to several other concepts. This relation is termed as ‘*hasAttributeOfInterest*’. The lightweight representation of this relation is shown in figure 5.12.

The above statement shows that the *Feature* concept is related to the concept *Particular* through the relation *hasAttributeOfInterest*. The concept *Particular* is a foundation concept from ULO which can represent any event or object. Different sub-concepts under the main concept *Feature* relate to concepts from different categories through the relation *hasAttributeOfInterest*.

Another inter-category relation of the *Feature* concept is *associatedTo*. This relation is defined to capture the association of the concept *PartFeature* with its corresponding *Part*. Further details about the inter-category relations of *Feature* concepts can be found in sections 6.4.1 to 6.4.3 of chapter 6.

5.3.6 Part Family’s Inter-Category Relations

It is possible to define a relation to assist in the capture of *PartFamily* criteria. A relation ‘*hasCriteria*’ has been introduced and applies at the more generic *Family* level. This relation is shown in figure 5.12. This relation relates *Family* to the criteria for the definition of that particular *Family*. The concept criteria can be any event or object. In the context of this thesis the criteria for the definition of *PartFamily*, *DesignPartFamily* and *ProductionPartFamily* concepts can also be captured through the relation *hasCriteria*.

5.3.6.1 With Manufacturing Method Concepts

The relation '*hasCriteria*' can assist the formalisation of *ProductionPartFamily* by capturing the criteria as follows:

- *ProductionPartFamily hasCriteria PartFamilyProductionMethod.*

The relation *hasCriteria* between *ProductionPartFamily* and *PartFamilyProductionMethod* is not shown in figure 5.12. This relation is not required to be shown because it is inherited by all the concepts subsumed under the concept *Family*.

Similarly, the relation *hasCriteria* can assist in the formalisation of the concept *DesignPartFamily*.

5.3.6.2 With Feature Concepts

Each *Part* is a combination of different *Features*. Therefore, a relation '*hasFeature*' is defined to capture this association. This relation is shown in figure 5.12.

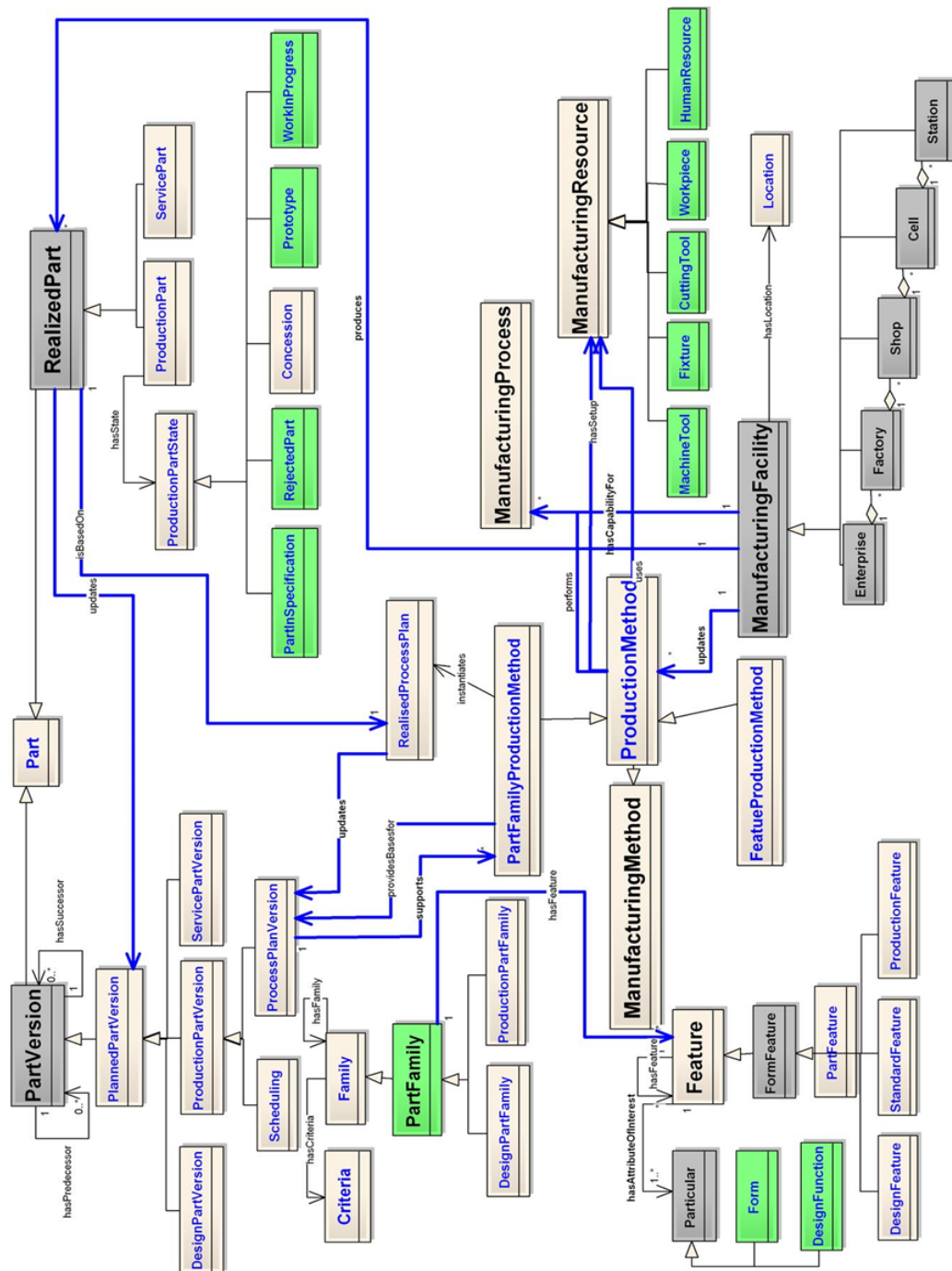
5.3.7 Manufacturing Method's Inter-Category Relations

5.3.7.1 With Manufacturing Process Concepts

It was observed during the industrial investigation and through the review of literature that the *ProductionMethods* could consist of a sequence of only the designated operation numbers. For example a *ProductionMethod* could be, OP10 followed by OP20 followed by OP30. In such cases, there is a requirement to capture the process being performed in an OP. Therefore, a relation '*performs*' has been defined to capture the description of processes being performed during the *ProductionMethod*. This relation is shown in figure 5.12.

5.4 Combined Lightweight Representation of the MCCO

A lightweight representation of the MCCO concepts and relations is presented in figure 5.12. It shows all the different categories of concepts, inter-category relations and intra-category relations involved in the MCCO. Some additional concepts and relations from the *ManufacturingMethod* category have not been shown in figure 5.12 and they are discussed in chapter 7.



The concepts in larger bold text in the black colour in figure 5.12 represent the categories of concepts and the bold blue lines represent the inter-category relations. The thinner black lines represent the intra-category relations. The concepts in normal text are intra category concepts belonging to their respective categories and the normal black lines show the intra-category relations.

The different colours of boxes in figure 5.12 are used to distinct the concepts that are (1-pink) original MCCO concepts, (2-green) concepts adapted from others and (3-grey) concepts adopted from others.

The semantics of the concepts are formally and unambiguously captured through formalisation in heavyweight logic. The formalisation is done in KFL. The next section reports the KFL formalisation of MCCO.

5.5 Heavyweight Formalisation of the MCCO

5.5.1 The Use of KFL for Heavyweight Formalisation

This section briefly illustrates the use of KFL to formally capture the semantics of concepts within the context of the MCCO. Detailed explanation on the use of the KFL is provided in appendix A1. The process of heavyweight formalisation in KFL consists of two steps, (1) the declaration of the required concepts, relations and functions and (2) axiomatization to formally capture the semantic of the concepts. The concept *Family* is used to illustrate the use of KFL for heavyweight formalisation.

5.5.1.1 Declaring Concepts, Relations and Function

The concepts in KFL are called ‘Properties’ and are declared using the directive ‘:Prop’. For example, a first step to formalise *Family* is to declare this concept in KFL as follows;

```
:Prop Family
:Inst Type
:sup Group
```

The directive ‘:Inst’ is used to declare the type of instantiation of concepts. Mostly the instantiations are either of kind ‘*Type*’ or ‘*MaterialRole*’ in KFL. Type represents the properties that always “stick” to their instances permanently. For example, an instance of a Property *Person* as type will always remain a person. *MaterialRole* in contrast, can come and go e.g. an instance of a *Person* as a *MaterialRole* can be Student at one time and a teacher at another. The directive ‘:sup’ is used to specify the super concept of a concept and thus builds a hierarchy. For example the super concept for *Family* is *Group* from the Upper Level Ontology (ULO).

In order to capture the criteria for the definition of a *Family* the concept *Criteria* is declared as follows;

```
:Prop Criteria
:Inst Type
:sup Particular
```


The super concept of *Criteria* i.e. *Particular* can either be an *Event* or an *Object*.

After declaring the concepts, the required relations are declared. For example, the relation '*hasCriteria*' is required to formalise the semantics of *Family* which is declared as follows;

```
:Rel    hasCriteria
:Inst   BinaryRel
:Sig     Family Criteria
```

According to the definition, a *Family* has to have a *Criteria*. The association of *Criteria* to *Family* is captured through the relation *hasCriteria*. The directive ':Rel' is used to declare the relations. The directive ':Inst' specifies the kind of relation and the number of the concepts involved e.g. a ':Inst BinaryRel' represents a relation over two concepts. If this relation was 'Inst SymmetricBinaryRel' then it would have referred to a relation with associative property over two concepts. The concepts involved are specified using the directive ':Sig' e.g. the concepts involved in *hasCriteria* are *Family* and *Criteria*.

In addition to concepts and relations, sometimes it is required to identify traits like mass, length and angle by declaring functions like kgs, mm and degrees. A function is not required to formalise the semantics of *Family* however, the declaration of functions is explained by declaring the function 'mm' which is used in the MCCO.

```
:Fun     mm
:Inst     UnaryFun
:Sig      RealNumber -> LinearDimension
```

The directive ':Fun' declares the function and the directive ':Inst' specifies the kind of function and the arity of number of concepts involved. The directive ':Sig' specifies the way a function works e.g. in the declaration of function 'mm' the signatures specifies that a *RealNumber* value in association to mm returns the *LinearDimension*. Where *RealNumber* is from ULO and *LinearDimension* is defined elsewhere in the MCCO.

5.5.1.2 Axiomatization

After the declaration of concepts, relations and functions in formal logic, the ontology still remains lightweight and is made heavyweight through axiomatization. Axiomatization refers to the defining of axioms to formally constrain the interpretations of the concepts. This equips the system with an ability to understand the semantics of concepts and thus helps in solving the semantic interoperability issues. Axioms in the context of KFL can either be 'inference rules' or 'integrity constraints (ICs)'. The

Inference rules, as the name suggests, are used to make inferences whereas the ICs formally capture and constrain the semantics of concepts. For example, to capture the definition of *Family* (section 5.2.8.1) formally, the following IC is written;

```
(=> (Family ?fam)
      (exists(?cri)
        (and (Criteria ?cri)
              (hasCriteria ?fam ?cri))))
:IC hard "The criteria for the Family should be defined"
```

The above axiom formally states “If there is a *Family* ?fam then a *Criteria* ?cri of that *Family* should exist and that should be related to the *Family* through the relation *hasCriteria*.” This implies that in order to assert a *Family* in the KB a *Criteria* has to be defined and related to it in accordance with the definition of *Family*. The directive “:IC hard” at the end of the axiom specifies that the IC cannot be violated during fact assertion in the KB and the asserted facts have to satisfy the condition specified by the hard IC. The other type of IC being used in the formalisation is ‘IC soft’ which can be violated but it generates a warning message on violation.

5.5.2 An example: Heavyweight Formalisation of PartFamily Concepts

The formalisation approach explained in the last section is applied to formalise the MCCO. In this section, the formalisation of a selected set of *PartFamily* concepts i.e. *Family*, *PartFamily*, and *ProductionPartFamily* is explained. The formalisation of *Family* has already been explained, therefore, the formalisation of other selected concepts is explained in the following sections.

5.5.2.1 Formalisation of the Concept PartFamily

Since the concept *PartFamily* is a sub concept of the concept *Family*, it inherits the formal semantics of the concept *Family*. However, the capture of the specific semantics of the concept *PartFamily* requires the declaration of the concept *PartFamily* and *Part* which is given below.

:Prop	PartFamily	:Prop	Part
:Inst	Type	:Inst	Type
:sup	Family	:sup	Object

According to the definition, *PartFamily* is a parametric *Part* that represents a group of *Parts*. This means that all the *Parts* belonging to a *PartFamily* should be represented by the same parametric *Part*. Therefore, the criteria for a *PartFamily* is to have a common parametric *Part* and the following IC captures this formally.

```
(=> (PartFamily ?pf)
      (exists (?pt)
        (and (Part ?pt)
              (hasCriteria ?pf ?pt))))
```

:IC hard “The parametric Part should be defined to satisfy the criteria for PartFamily”

The above IC states that if there is a *PartFamily* ?pf then there has to exist a *Part* ?pt as the criteria of the *PartFamily* and related to the *PartFamily* by relation *hasCriteria*.

5.5.2.2 Formalisation of the Concept ProductionPartFamily

According to the definition, the criteria for a *ProductionPartFamily* is to have a common *ProductionMethod*. This common *ProductionMethod* can be referred to as *PartFamilyProductionMethod*. Therefore, the new concepts used are, *ProductionPartFamily*, *PartFamilyProductionMethod* which are declared as follows;

:Prop	ProductionPartFamily	:Prop	PartFamilyProductionMethod
:Inst	Type	:Inst	Type
:sup	PartFamily	:sup	ProductionMethod

The definition of *ProductionPartFamily* are formally captured through the following axiom.

```
(=> (ProductionPartFamily ?prdpf)
(exists (?pfpm)
  (and (PartFamilyProductionMethod ?pfpm)
    (hasCriteria ?prdpf ?pfpm))))
:IC hard “The PartFamilyProductionMethod ?pfpm should be defined for the concept
ProductionPartFamily”
```

The above axioms formally captures that If there is a *ProductionPartFamily* ?prdpf then there has to exist a *PartFamilyProductionMethod* ?pfpm as the criteria for *ProductionPartFamily* and related to *ProductionPartFamily* by the relation *hasCriteria*.

The same methodology applies to the formalisation of semantics of *DesignPartFamily* and all other MCCO concepts and relations. The formalisation of concepts and relations may also involve inference rules, which are visited in more detail in chapter 7. The heavyweight formalisation of all the concept and relations in MCCO can be referred to in appendix A2 at the end of this thesis.

5.6 Summary

A core set of manufacturing concepts has been identified in the form of a Manufacturing Core Concepts Ontology (MCCO). The MCCO is neither a purely design ontology nor a standalone production ontology but it is a Manufacturing ontology which can support knowledge sharing across product design and production domains. Although the MCCO is not a production ontology, a considerable number of MCCO concepts are oriented towards production. This is because the ontology is designed to support the capture and sharing of production knowledge into design. The production oriented concepts in the MCCO are not specific and are generic enough to support the development of ontologies for any production application. This means that

the MCCO can be extended to include generic concepts for other product lifecycle disciplines like design, operation and disposal. The present version of the MCCO can be used as a common semantic base for developing specific product design and production ontologies and for knowledge sharing across these domains.

The formalised concepts eliminate ambiguities in interpretation of manufacturing concepts and thus help the knowledge system to relate concepts with similar semantics between product design and production concepts. The MCCO provides a verifiable semantic base for sharing knowledge between product design and production domains through the formally captured meanings of concepts.

The concepts and relations in the MCCO have varying depths of meaning i.e. some are generic while some are more specific in their meaning and the next chapter presents a method to capture this varying depths of meaning of different concepts.

6 Specializing Concepts to Capture the Varying Depths of Meaning

6.1 Introduction

A requirement for capturing varying depths of meaning of different concepts has already been identified in section 4.4 of chapter 4 and this chapter details the method developed to meet that requirement. An approach to capture the varying depths of meaning of concepts by specializing them at different levels has been proposed. The chapter is organized as follows. Section 6.2 briefly explains the varying depths of meaning of concepts within the Manufacturing Core Concepts Ontology (MCCO). Section 6.3 presents an overview of the different levels of specialization. Section 6.4 presents an example of specialization of *Feature* concepts. Section 6.4 also presents the formalization of *Feature* concepts at different levels of specialization to capture the variations in depths of meaning.

This chapter also addresses the questions (raised in points 2 and 3 of section 4.3 of chapter 4) regarding the development of specialized design and production ontologies and the route to knowledge sharing between them. The capability of the MCCO to support the development of specialized product design and production domain ontologies is discussed in section 6.5 and the provision of a route to knowledge sharing between these two domains has been explained in 6.6. Section 6.7 presents a summary of the chapter.

6.2 Variation of Depths of Meaning within the MCCO

The main case argued in this chapter is that the variations in depths of meaning of concepts can be captured by specializing them through multiple levels.

Similar to the variations in meaning of concepts initially reported in section 4.4 of chapter 4, the set of concepts and relations included in the MCCO have variations in their depths of meaning with respect to their applicability. Examples of these are:

- Concepts generic to any domain e.g. *Feature*, *Function*, *Form*, *hasCriteria*, *hasAttributeOfInterest*, and *uses*.
- Concepts generic to any of the product lifecycle domains e.g. *Part*, *PartVersion*, *PartFamily*, and *PartFeature*,
- Concepts specific to the individual product lifecycle domain. For example *DesignFeature*, *DesignFunction*, and *DesignPartFamily* are specific to design

domain and concepts like *ProductionFeature*, *ProductionPartFamily*, *MachineTool*, and *CuttingTool* are specific to the production domain.

This leads to the identification of different ranges or levels of concepts in the MCCO with respect to the variations in their depths of meaning.

6.3 An Overview of Specialization Levels

The general idea of specialization is not new. Lee and Suh (2008) proposed an ontology based multi layer knowledge framework to capture varying degrees of specializations of knowledge. Oberle et al (2007) used an approach of specialization for developing the SWIntO ontology from the DOLCE and SUMO foundation ontologies. SWIntO could be specialized further to develop more specialized ontologies. The PSL (ISO-18629) ontology also has multiple levels of specialization e.g. PSL Core, PSL outer Core and PSL extensions. The specialization in PSL is done by treating ontologies as theories and new theories are extended from core theories. Moreover, PSL doesn't identify concepts or taxonomies of concepts and all entities in the ontology are either relations, functions or constants. In this respect, the specialization approach proposed in this thesis is slightly different because in this approach a taxonomical perspective as well as a semantic conformance viewpoint is considered.

The specialization in this thesis is not just about a taxonomy of concepts where generic concepts are super concepts of the relatively specific concepts. It is also about the consistency and conformance of the inherited semantics of concepts and relations. The novelty of the research work reported here is to explicitly define the levels of specializations where each level captures a gradual variation in the depths of meaning of concepts.

Three levels of specialization of concepts have been proposed to capture the variation in depths of meaning of manufacturing concepts. Each level represents a certain degree of specialization which supports the capture of a certain depth of meaning of concepts. Figure 6.1 shows the specialization levels in the MCCO. The figure also shows the development of application specific domain ontologies using the MCCO and the interoperability across them. The three levels are named as:

1. Generic Core Concepts Level
2. Product Lifecycle Generic Level
- 3(a) Design Core Concepts Level
- (b) Production Core Concepts Level

According to the proposed approach, concepts in each specialized level have an increased degree of specialization as compared to the previous level. As the

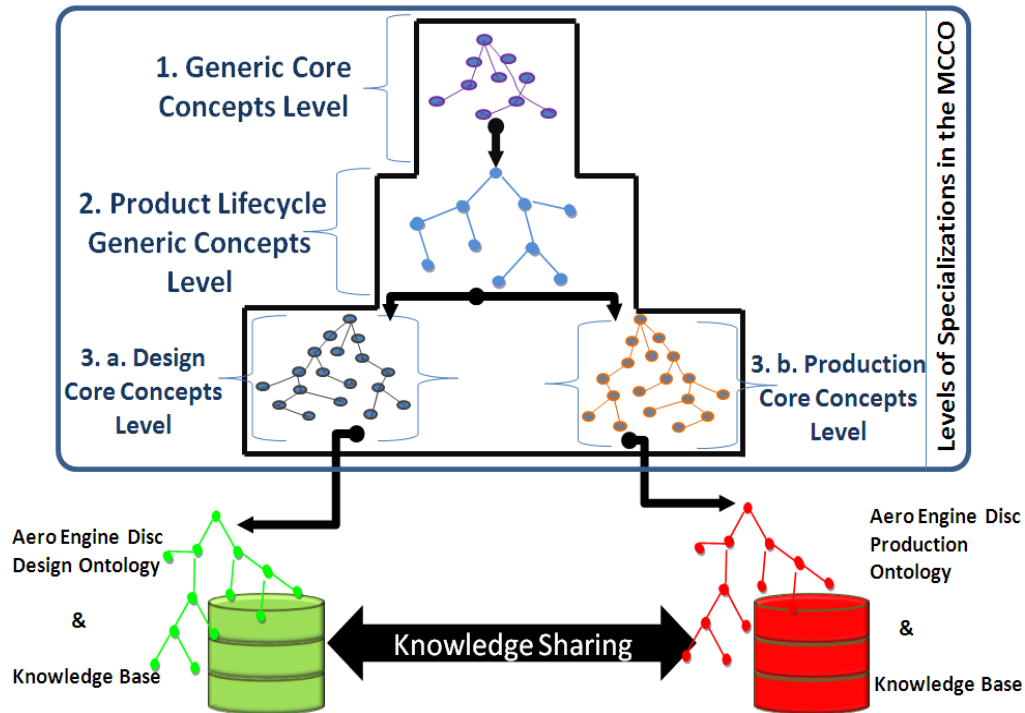


Figure 6.1: Schematic view of the specialization levels in the MCCO and how they help develop application specific ontologies.

specialization level gets closer to the application specific domains, the semantics of concepts also get closer to the application domain. Each level can potentially act as a semantic base for the concepts specialized further from it. Thus, these levels with gradually increasing degree of specializations support the capture of the varying depths of meaning and provide a more effective route to knowledge sharing by providing concepts that are semantically sound and closer to the product design and production domains.

6.3.1 Generic Core Concepts Level

This level identifies concepts from the foundation ontology and specializes them at a useful level for manufacturing. Thus, these concepts are more specific in conceptualization than the foundational concepts. For example, the foundation concept *Particular* can be specialized into generic core concepts *Feature*, *Family*, and *Part*. The generic core concepts can be used by any domain due to their generic semantics. The semantic base provided by generic core concepts can be used to develop product lifecycle core concepts.

Moreover, the generic core concepts level can be used to connect product lifecycle domains ranging from product design and production through to business, marketing

and finance. However, in the context of this work, domains other than the product design and production are beyond the scope.

6.3.2 Product Lifecycle Generic Core Concepts Level

This level enables capturing the meaning of concepts generic to the product lifecycle. This means that product lifecycle level core concepts can only be used within the product lifecycle domains. For example they can be used in design, production, operation, or disposal domains but not for domains like business, finance, and marketing. Examples of concepts at this level are *PartFeature*, *Product*, *Part*, *ManufacturingFacility*, *PartFamily*, *GeometricTolerance*, and *PlannedPart*. This level contains the core set of concepts specialized from the generic core concepts. Because concepts at this level are closer to product design and production concepts, they can provide a more relevant and effective route for knowledge sharing between product design and production domains.

6.3.3 Product Design and Production Core Concepts Levels

This level enables capturing the depths of meaning of concepts specific to individual product lifecycle domains. It is understood that the two domains considered in this thesis are the product design and production domains. Concepts at this level are specialized from product lifecycle generic core concepts. Core product design and production concepts can support the development of application-specific product design and production ontologies. It is important to mention here that concepts at this level are part of the MCCO and are not part of the specific design and production ontologies.

6.3.3.1 Product Design Core Concepts Level

Concepts specific to product design are defined at this level. Examples of design core concepts are *DesignFeature*, *DesignFunction*, and *DesignPartFamily*.

These concepts can be used to develop customized product design ontologies for specific applications by further specializing or even adding new concepts. For instance, the concept *DesignFeature* can be specialized for an aero engine disc design ontology into the concepts *Diaphragm*, *BoltHole*, *Cob*, and *BalanceLand* for representing different *DesignFeatures* of the disc as shown earlier in figure 3.2. Thus, application specific design ontologies (e.g. an aero engine disc design ontology) can be built using the design core concepts of the MCCO.

6.3.3.2 Production Core Concepts Level

Concepts specific to the production domain are defined at this level. Examples of production core concepts are *ProductionFeature*, *ProductionMethod*, and *ProductionPartFamily*. These concepts can be used to develop customized production ontologies for specific applications by further specializing or even adding new concepts. For instance *ProductionFeature* can be specialized for an aero engine production ontology into concepts representing *WebProfile*, *Rim*, *WebHole*, and *Hub* features of a disc as shown earlier in figure 3.4.

Thus, production core concepts can support the development of ontologies that capture knowledge for application-specific production ontologies, e.g. an aero engine disc production ontology and knowledge as shown in figure 6.1.

6.4 An example: Specialization Levels Explained using Feature Concepts

The essence of specialization levels and their ability to capture varying depths of meaning of concepts is elaborated by taking one key concept and showing its journey through various levels. Feature concepts have been selected here for this purpose.

Feature concepts are the main set of core concepts in the MCCO which facilitate relating product design and production domains. They are relatively simple to explain with relatively simple relations and axioms. They have the gradual variations in depths of meaning flowing through all the specified levels of specialization. Feature concepts act as building blocks for the design and production part families. They bring together manufacturing information from different modules of manufacturing. Moreover, the ontology is more developed from a feature point of view. These are the reasons for choosing *Feature* concepts to explain the capture of varying depths of meaning of concepts using different levels of specializations.

Figure 6.2 shows examples of some possible variations in the meanings of the concept *Feature*. It also shows the journey of the concept *Feature* through the three proposed levels of specializations for representing those variations in meanings. As shown in figure 6.2, the interpretation of *Feature* varies from being very generic to being specific to product design and production. For example, a *Feature* can be generically interpreted to be a smile, running fast, a shape, or even a skin colour where the interpretation is independent of any specific domain. However, a *Feature* can also be interpreted for a specific and restricted domain only e.g. to only represent the design features like 'joining feature' and 'drilling feature' as illustrated in figure 6.2.

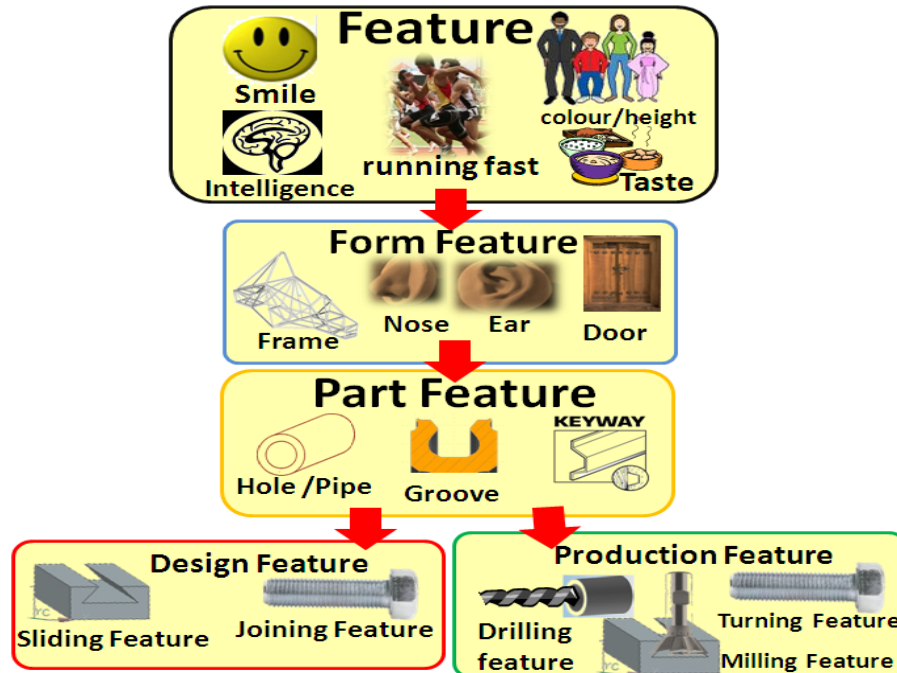


Figure 6.2: Varying specializations of feature concepts in everyday life

Sections 6.4.1 to 6.4.3 explain how the variations in meaning of *Feature* concepts are formally captured using different levels of specialization. In these sections the *Feature* concepts and their different levels of specialization to capture the gradual variations in depths of meaning are discussed and defined.

6.4.1 Generic Level Feature Concepts

Two *Feature* concepts have been identified and defined at the generic level. A discussion on the definition of these concepts and their formalization is presented in this section.

6.4.1.1 'Feature' Concept

At the generic core concepts level *Feature* can be interpreted independent of any viewpoint. In that context, the definition of *Feature* as “a prominent attribute or aspect of something” (Webster's online Dictionary 2011; WordNet 2010) establishes the essence of *Feature* to have an attribute of interest. According to this, *Feature* is a fairly generic term which can be the dark hair of a person, ability of a person to run fast, singing, skin colour, size and shape of building, shape of an article, etc as shown in figure 6.2. Therefore, the concept *Feature* at a generic level has been informally defined as; “*Feature* is anything having a particular attribute of interest”

6.4.1.2 Formalization of 'Feature' Concept

The UML diagram in figure 6.3 shows the lightweight formalization of the concept *Feature*. Figure 6.3 shows that a *Feature* has a *Particular* attribute of interest. The

concept *Feature* is related to the *Particular* by the relation *hasAttributeOfInterest*. The one-to-many relation (1 1..*) from *Feature* to *Particular* in figure 6.3 shows that a *Feature* can have one or more *Particular* attributes of interest. The concept *Particular* is a foundational concept from ULO. The attribute of interest of a *Feature* can either be an event or an object.

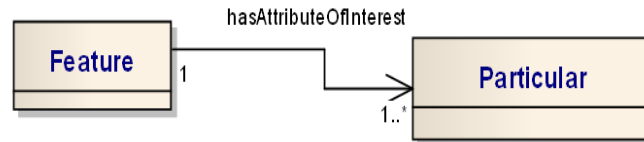


Figure 6.3: Lightweight representation of feature concept

Following the use of KFL as described in section 5.5.1, the concept *Feature* and its required relation *hasAttributeOfInterest* are represented as follows;

:Prop Feature	: Rel hasAttributeOfInterest
:Inst Type	:Inst BinaryRel
:sup Object	:Sig Feature Particular

The definition of *Feature* is then, formally captured through the following axiom.

```

(=> (Feature ?feature)
    (exists(?p)
      (and (Particular ?p)
            (hasAttributeOfInterest ?f ?p))))
:IC hard "Feature requires an attribute of interest"
  
```

The above axiom states “If there is a *Feature* ‘?feature’ then a *Particular* ‘?p’ related to that *Feature* through the relations *hasAttributeOfInterest* has to exist. This implies that in order to assert a *Feature* in the KB its *Particular* attribute of interest had to be defined and related to it in accordance with the definition of the concept *Feature*. This rule is placed as a hard IC in the MCCO which prevents the assertion of any facts in violation of the definition of *Feature*. The firing of the ICs when the definition of *Feature* is violated shows that the system understands the semantics of the concepts. This is experimentally investigated in chapter 8 of the thesis.

6.4.1.3 FormFeature Concept

The kind of *Features* of interest in this thesis will always have a form. This type of *Feature* requires the capture of the depth of its meaning involving an associated form. In order to capture this level of depth in meaning, a more specialized concept *FormFeature*, is identified. Examples of *FormFeature* can be nose, door, window and frame. The conceptualization of *FormFeature* is not specific to manufacturing and is generic to any domain that makes it a generic level core concept. The concept *FormFeature* is also found in previous literature. Semere et al (2007), for example, had *FormFeature* as one of the main concepts in their machining ontology. Semere et

al (2007) defined *FormFeature* to represent a manufacturing product where 'geometry', 'tolerances', and other technical data were the required attributes of *FormFeature*. *FormFeature* has also been defined in the Core Product Model (CPM) (Fenves et al. 2006), to have the geometry and material. In this thesis, the concept *FormFeature* is required to be able to capture the Form irrespective of the material. This helps in keeping the core concepts flexible and enables the capture of information at a generic level. Therefore, *FormFeature* is informally defined as follows:

"The concept *FormFeature* represents a *Feature* which has a *Form* as its required attribute of interest"

6.4.1.4 Formalization of FormFeature

The lightweight formalisation of the semantics of *FormFeature* is shown in figure 6.4. The *Feature* concept subsumes the *FormFeature* concept as shown in figure 6.4. The relation *hasAttributeOfInterest* relates a *FormFeature* to its *Form*. The relation *hasAttributeOfInterest* is not required to be shown in figure 6.4 because it is inherited from the *Feature* concept.

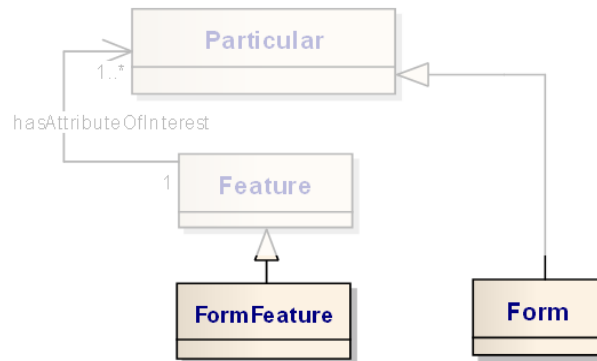


Figure 6.4:Lightweight representation of FormFeature

The heavyweight formalization of *FormFeature* follows the same methodology as explained for *Feature*. This involves declaring of the required concept *FormFeature* and *Form* in KFL and the capture of semantics through the following axiom;

```

(=> (FormFeature ?ffeature)
    (exists (?form)
      (and (Form ?form)
            (FormFeature ?ffeature)
            (hasAttributeOfInterest ?ffeature ?form))))
:IC hard "Every FormFeature has a form"

```

The above axiom states "If there is a *FormFeature* '*?ffeature*' then there has to exist a *Form* '*?form*' related to the '*?ffeature*' by the relation *hasAttributeOfInterest*". The above axiom captures the definition of the concept *FormFeature* and constrains the

existence of *FormFeature* with its required attribute of interest i.e. *Form*. The above axiom prevents the assertion of any *FormFeature* fact in the KB without first defining its *Form* as an attribute of interest.

6.4.2 Product Lifecycle Generic Feature Concept

In the context of parts, the concept *FormFeature* should represent a certain portion of a component on a part e.g. groove on a disc, hole in a pipe, and keyway in a shaft as shown in figure 6.2. Therefore, another more specialized concept needs to be identified to capture the semantics that refer to the part on which the *FormFeature* exists. The concept defined for that purpose is termed as *PartFeature*. A *PartFeature* may exist in different product lifecycle domains like design, production, operation and disposal. Therefore, the semantics of *PartFeature* are needed to be generic enough to be suitable for any of the product lifecycle domains.

In this context, a number of definitions of *Feature* are available in international standards (ISO-10303-APs1101, 1130, 207, ISO 13584). The more relevant one is that given by ISO-10303 which defined a *Feature* as: “local geometric configuration belonging to a product shape description, having significance in some application context associated with the product model”(ISO-10303-108 2005). Based on the industrial study and the adopted specialization, the informal definition of the concept *PartFeature* is stated as “The concept *PartFeature* represents a *FormFeature* associated to a *Part*.”

6.4.2.1 Formalization of the Concept ‘PartFeature’

Figure 6.5 shows the lightweight formalization of the concept *PartFeature*. It has been subsumed under the concept *FormFeature*. The relation *associatedTo* relates *Part* to *PartFeature*.

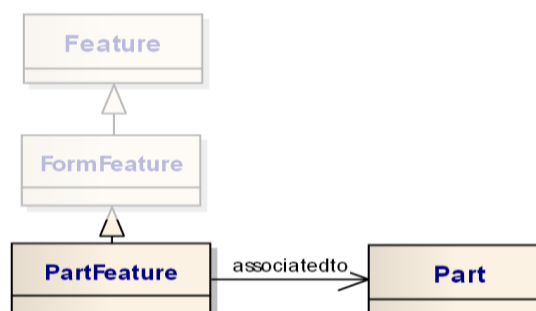


Figure 6.5: Lightweight Product Lifecycle level specialization through Product feature concept

Note that instead of *hasAttributeOfInterest* the relation *associatedTo* is used. This is because *hasAttributeOfInterest* is only used to refer to the required attributes of features. However, a *Part* is not placed as a required attribute and rather an optional

attribute for a *PartFeature*. This is because there can be conditions where it required to capture the Form of a *PartFeature* without knowing about its associated *Part*. For example, the information about a hole and its form can be captured irrespective of the hole being associated to a block, shaft, or disc. That is why the association of a part to a Feature is not placed as a hard constraint. This method provides the flexibility to capture knowledge with or without an associated *Part*.

For the heavyweight formalisation of *PartFeature*, the relation *associatedTo* is declared whereas the concept *Part* is declared elsewhere. The semantics of *PartFeature* are then captured through the following axiom.

```
(=> (PartFeature ?Pfeature)
      (exists (?P)
        (and (Part ?P)
              (associatedTo ?Pfeature ?P))))
:IC soft "Every Part feature has an associated Part"
```

The above axiom states that for a *PartFeature* ‘?Pfeature’ to exist, an associated *Part* ‘?P’ may also exist. The directive ‘IC soft’, will result in generating a warning message when a *PartFeature* is asserted in KB without its associated *Part* but will not cancel the assertion.

The semantics of the concept *PartFeature* make it suitable for any of the product lifecycle domains. Therefore, it can provide a basis for defining and relating the more specialized *Feature* concepts for the design and production domains.

6.4.3 Design and Production level Feature Concepts

The concept *PartFeature* is significant for both product design and production domains. The depths of meaning involving the required level of detail can be captured in the proposed specialization levels. The concepts which can be used to capture the depths of meaning need to be defined at design level and production level specializations.

6.4.3.1 Design Level Feature Core Concept

The conceptualization of *DesignFeature* is defined with respect to the functional requirements of the *PartFeature*. Therefore, a *DesignFeature* is defined as “a *PartFeature* having a *DesignFunction* as a defining attribute of interest”. Examples of *DesignFeature* are *SlidingFeature* and *JoiningFeature* as shown in figure 6.2.

6.4.3.2 Formalization of 'DesignFeature'

The lightweight formalization of a *DesignFeature* is given in figure 6.6 represented by highlighted concepts. The relation *hasAttributeOfInterest* relates the concept *DesignFeature* to its required attribute represented by the concept *DesignFunction*. The relation *hasAttributeOfInterest* is not shown in the figure between the concepts *DesignFeature* and *DesignFunction* because it is inherited from the concept *Feature* and does not need to be shown.

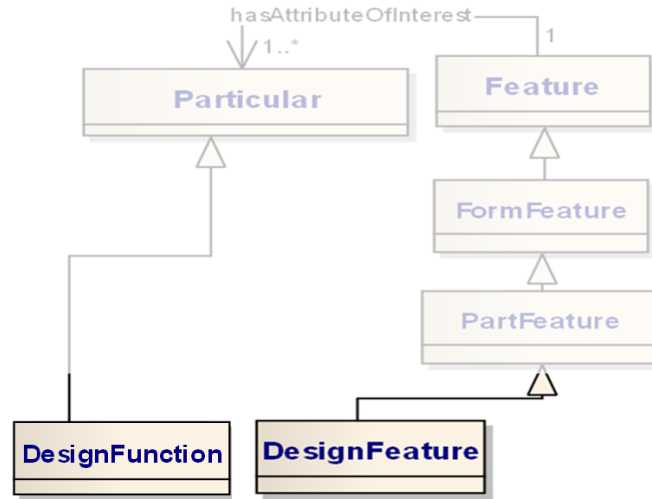


Figure 6.6: Lightweight formalization of Design level specialization of Feature as DesignFeature

Two types of axioms, i.e. defining axioms and controlling axioms, are used in the heavyweight formalization of the *DesignFeature*. The defining Axiom below captures formally the definition of *DesignFeature*.

```
(=> (DesignFeature ?df)
  (exists (?dfunction)
    (and (DesignFunction ?dfunction)
      (hasAttributeOfInterest ?df ?dfunction))))
:IC hard "A DesignFunction has to be defined as an Attribute of Interest for the DesignFeature"
```

The above IC captures that for a *DesignFeature* ?df there has to exist a *DesignFunction* ?dfunction as the attribute of interest of *DesignFeature* and related to it by the relation *hasAttributeOfInterest*.

The controlling axiom makes the fact assertion foolproof by not allowing a *DesignFeature* to be asserted with a *ProductionMethod* as its attribute of interest. The false fact assertion can happen due to confusion of *DesignFeature* with *ProductionFeature* when asserting several facts about different features. The IC for that is as follows.

```
(=> (and (DesignFeature ?df)
  (FeatureProductionMethod ?fpm))
  (not (hasAttributeOfInterest ?df ?fpm)))
:IC hard "FeatureProductionMethod does not apply to DesignFeature"
```

This IC means that a *DesignFeature* ?df cannot be asserted with a *FeatureProductionMethod* ?fpm as its attribute of interest. In this way this IC controls and prevents any false assertions.

6.4.3.3 Production Level Feature Core Concepts

Based on the understanding from the industrial study and the ISO standard (ISO-10303-Ap224), a *ProductionFeature* can be defined with respect to the method of its production. Therefore, the concept *ProductionFeature* is defined informally as; “a *PartFeature* having a *ProductionMethod* as its defining attribute of interest”

6.4.3.4 Formalization of ‘ProductionFeature’

The lightweight UML representation of *ProductionFeature* is represented by the highlighted concepts in figure 6.7. The relation *hasAttributeOfInterest* is reused. A *FeatureProductionMethod* is a required attribute of interest for a *ProductionFeature*.

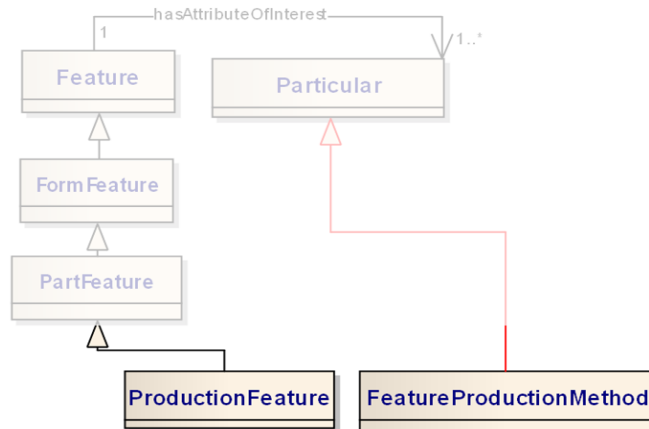


Figure 6.7: Lightweight formalization of Production level specialization of Feature as *ProductionFeature*

Like *DesignFeature*, two types of axioms are used for the heavyweight formalization of the *ProductionFeature*. The defining axiom, given below, formally captures the definition of *ProductionFeature*.

```

(=> (ProductionFeature ?pf)
  (exists (?fpm)
    (and (FeatureProductionMethod ?fpm)
      (hasAttributeOfInterest ?pf ?fpm))))
:IC hard "A FeatureProductionMethod may be defined for the ProductionFeature"

```

The above axioms reads that for a *ProductionFeature* ?pf there has to exist a *FeatureProductionMethod* ?fpm as its attribute of interest. The controlling axiom below, helps avoid any false assertions that;

```

(=> (and (ProductionFeature ?prodf)
  (DesignFunction ?dfun))
  (not (hasAttributeOfInterest ?prodf ?dfun)))
:IC hard "Function does not apply to a production feature"

```


The last axiom states that a *ProductionFeature* ?pf cannot be asserted with a *DesignFunction* ?dfun as its attribute of interest.

6.5 Development of Specialized Ontologies using Feature Concepts

The MCCO provides the semantic base that can support the development of application specific product design and production ontologies. The common semantic base provided by the MCCO also provides a route for knowledge sharing across specific product design and production domains. This section reports the ability of the MCCO to support the development of application-specific ontologies.

The first figure of this chapter i.e. figures 6.1, showed a conceptual view of application specific ontologies and KBs being developed using the MCCO. Following that same view, the *Feature* concepts from the MCCO are explored for their ability to support the development of application-specific product design and production ontologies as shown in figure 6.8. As examples of application-specific ontologies, an *AeroEngineDiscDesignOntology* and an *AeroEngineDiscProductionOntology* are developed as shown in figure 6.8. These application specific ontologies are then used to capture the design and production views of the studied disc presented earlier in figure 3.6. These views are captured as instances of the application specific ontologies as shown in the bottom portion of figure 6.8.

The top portion of figure shows the two core concepts from the MCCO i.e. *DesignFeature* and *ProductionFeature*. The central portion of the figure 6.8 demonstrates how these core concepts have supported the development of application specific ontologies i.e. the *AeroEngineDiscDesignOntology* and the *AeroEngineDiscProductionOntology*. Figure 6.8 shows the lightweight UML views of these ontologies. The detailed heavyweight formalization of application specific ontologies can be found in section A2.7 of appendix A2. Figure 6.8 illustrates that the core concept *DesignFeature* has been used to develop an *AeroEngineDiscDesignOntology* and similarly the core concept *ProductionFeature* can be used to develop the *AeroEngineDiscProductionOntology*.

The bottom portion of figure 6.8 shows how the concepts from application-specific ontologies are instantiated to capture the design and production views of the studied disc. The use of the core concepts from the MCCO to develop the application specific ontologies means that the application-specific ontologies agree on the MCCO. The commitment to MCCO provides semantic consistency and a route to knowledge sharing.

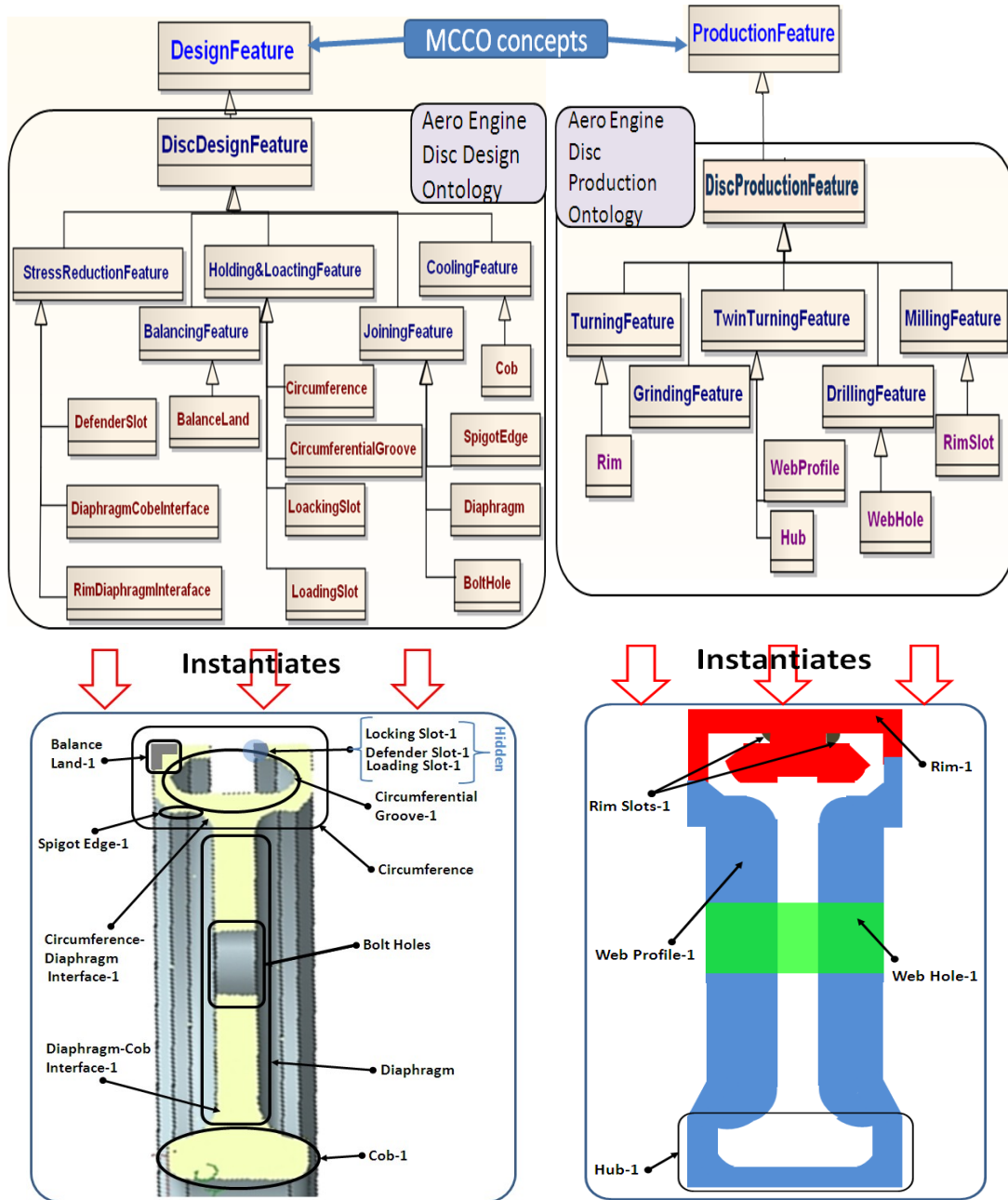


Figure 6.8: Lightweight view of application specific ontologies developed from the MCCO and the instantiated disc design and production views of application specific ontologies.

While this example focuses on *Feature* concepts, other concepts (i.e. *Form*, *Part*, *Function*, and *ProductionMethod* and other related concepts) and relations (i.e. *hasAttributeOfInterest*, and *associatedTo* and other related relations) from the MCCO have also been used. Indirectly through the relations present in the MCCO semantic integrity of the whole of the MCCO is inherited into the application-specific ontologies.

6.6 Route to Knowledge Sharing Through the Feature Concepts

Design and production features from the specific design and production ontologies can be related to share knowledge through the *Feature* concepts from the MCCO. For

instance, different disc design features can be related to their corresponding disc production features through *PartFeature*, *FormFeature* and the *Feature* concepts from the MCCO.

The immediate underlying level which can establish the link between disc design and production features is the product lifecycle *Feature* concept i.e. *PartFeature*. If different design and production features are related to the same *Part*, they can be linked to share knowledge about the *Part*. However, if the knowledge regarding a particular feature is required, then as understood from section 3.3.2 of chapter 3, design and production features can be related through the overlapping portions of their *Forms*. This is illustrated with the help of an example.

6.6.1.1 An Example of Relating Design and Production Feature(s)

Consider a case where the production consequences of changing the design of design feature 'CircumferentialGroove' are to be found. The required production consequences can be found if the *ProductionFeature* relevant to the *CircumferentialGroove* is found.

This is possible by identifying the production feature(s) that encompasses the form of design feature in question. But, the forms of design features are different from the forms of production features. However, portions of forms of design and production features can overlap with each other. In this case the form of *ProductionFeature* 'Rim' overlaps with the form of *DesignFeature* *CircumferentialGroove* as well as the form of *DesignFeature* *BalanceLand* as depicted in figure 6.9.

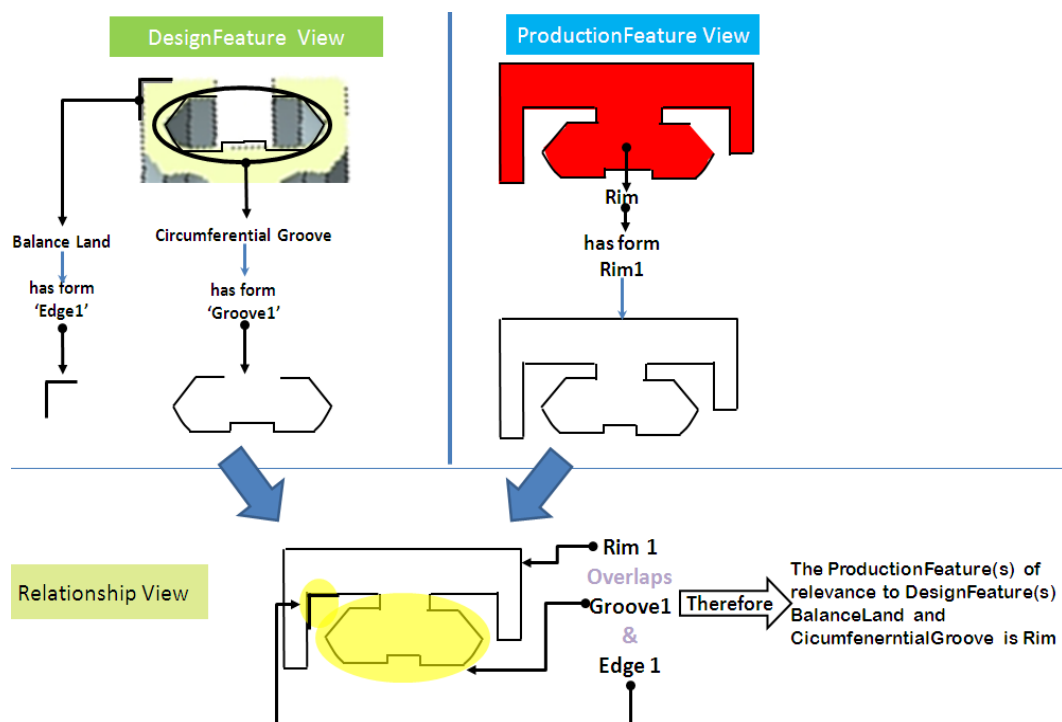


Figure 6.9: Relating design and production features through their overlapping forms

The figure shows that even through the forms of design and production features are different, they can still be related through the overlapping portions of their forms. However, the discussion is limited to *CircumferentialGroove* because this is the *DesignFeature* in question. Therefore, based on the relation established between *CircumferentialGroove* and *Rim* through their overlapping forms, it is understood that the production feature relevant to *CircumferentialGroove* is *Rim*. Therefore, the knowledge associated to *Rim* is relevant for the design of *CircumferentialGroove*.

It is understood that the production knowledge for production features can be captured in their corresponding production methods. Based on the established relation between '*CircumferentialGroove*' and '*Rim*', the knowledge associated to *Rim* can be shared with the relevant design feature(s) *CircumferentialGroove* and *BalanceLand*.

Examples of the knowledge associated to *Rim* that can be fed back to the design are "1. The value of the neck width should be greater than the diameters of the available cutting tools", "2. The groove angle should be greater than 25 deg for the groove to be machined with available tooling" and "3. The value of neck width of *Rim* groove should be greater than 10mm for machining with available tooling". The capture of this knowledge and the route to sharing this knowledge to design is experimentally verified in chapter 8 of the thesis.

In this example, the knowledge sharing has been explored at the feature level using the *Feature* concepts from the MCCO. However, the knowledge sharing can also be explored at the part level using the *PartFamily* concepts from the MCCO. The semantics of the *PartFamily* concepts have been captured but knowledge sharing has not been tested at the part level in this thesis.

6.7 Summary

Chapter 6 has proposed an approach to capture the varying depths of meaning of different concepts by specialising them at different levels of specialisations. Based on that approach, a representation of different *Feature* concepts is shown in figure 6.10. The UML figure presents the lightweight formalisation of *Feature* concepts at different levels of specialisations. This approach enables knowledge systems to understand the variations in the depths of meaning of concepts.

The chapter has also illustrated the capability of the MCCO to support the development of specialized product design and production domain ontologies and the provision of a route to knowledge sharing between product design and production domains through features.

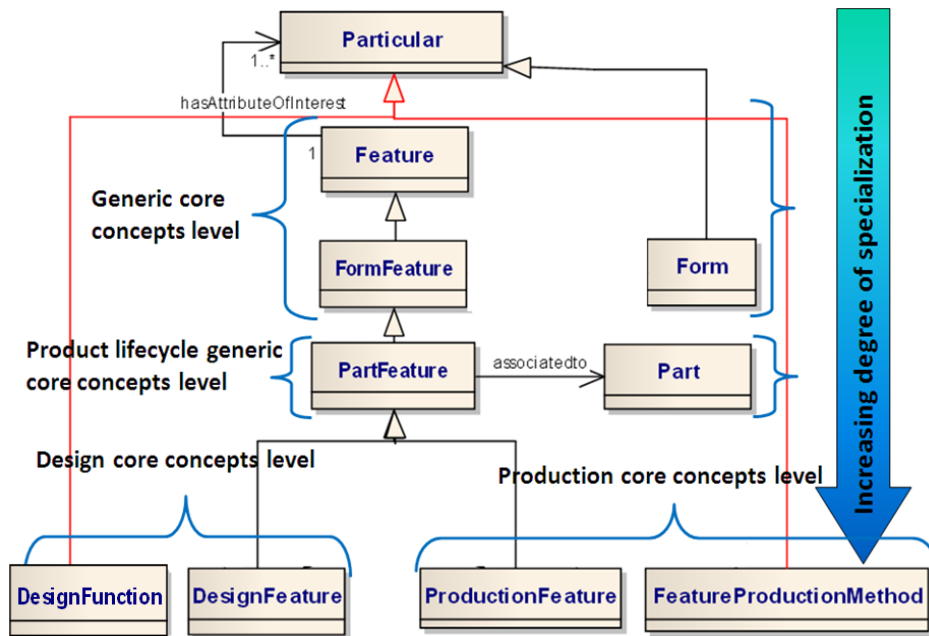


Figure 6.10: Feature concepts across specializations level, lightweight representation

Not only the concepts have different levels but also the production knowledge i.e. production methods for features and parts also need to be captured and reasoned about at different levels of abstraction (section 4.5 of chapter 4). The next chapter proposes a structure of concepts and relations to capture and reason about Meta level production methods.

7 The use of ‘clabject’ to capture and reason about Meta level Production Methods

7.1 Introduction

This chapter is developed against the research questions raised in section 4.5 of chapter 4 regarding the requirements to capture and reason about production methods at *Meta level knowledge*. This chapter further explains the requirements to capture and reason about production methods at Meta as well as individual levels of knowledge abstraction. The chapter introduces how ‘clabjects’ (explained in section 7.3.2.1) have been used to capture and reason about knowledge at the *Meta level knowledge* in relation to the production methods.

The chapter is organized as follows. Section 7.2 explains the requirements to capture and reason about production methods at both Meta and Individual levels of knowledge. Section 7.3 details how the production methods for features and part families can be captured and reasoned about at the *Meta level knowledge*. Section 7.3 thoroughly illustrates the lightweight and heavyweight formalisation of production methods which enables the capture of and reasoning about production methods at *Meta level knowledge*. Section 7.4 presents a summary of the chapter.

7.2 Requirements for Multiple Levels of Knowledge Abstractions

A requirement to capture and reason about *Meta level production knowledge* was identified in section 4.5 of chapter 4. That requirement is there to enable the capture of production engineers’ knowledge to support decision making during the early phases of product development and to know whether a part or a feature is manufacturable with existing production methods during early stages.

The more detailed examples of Meta and individual level production methods were presented in figures 4.3 and 4.4 in section 4.5.2 of chapter 4. Figure 7.1 here presents an informal representation of production methods for features and parts at multiple levels of knowledge abstraction. The Figure illustrates the subdivision of *ProductionMethod* into *FeatureProductionMethod* on the right and *PartProductionMethod* on the left. Both are further sub divided into Meta level and individual level production methods where the individual level production methods are instantiated from Meta level production methods.

On the left hand side in figure 7.1, examples of two different Meta level part family production methods i.e. *StandAloneDiscFamilyProductionMethod* and

ProjectedDiscFamilyProductionMethod are shown. The figure also shows that the process plans at the individual level knowledge are instances of the meta level part family production methods, as shown by the red arrows in figure 7.1.

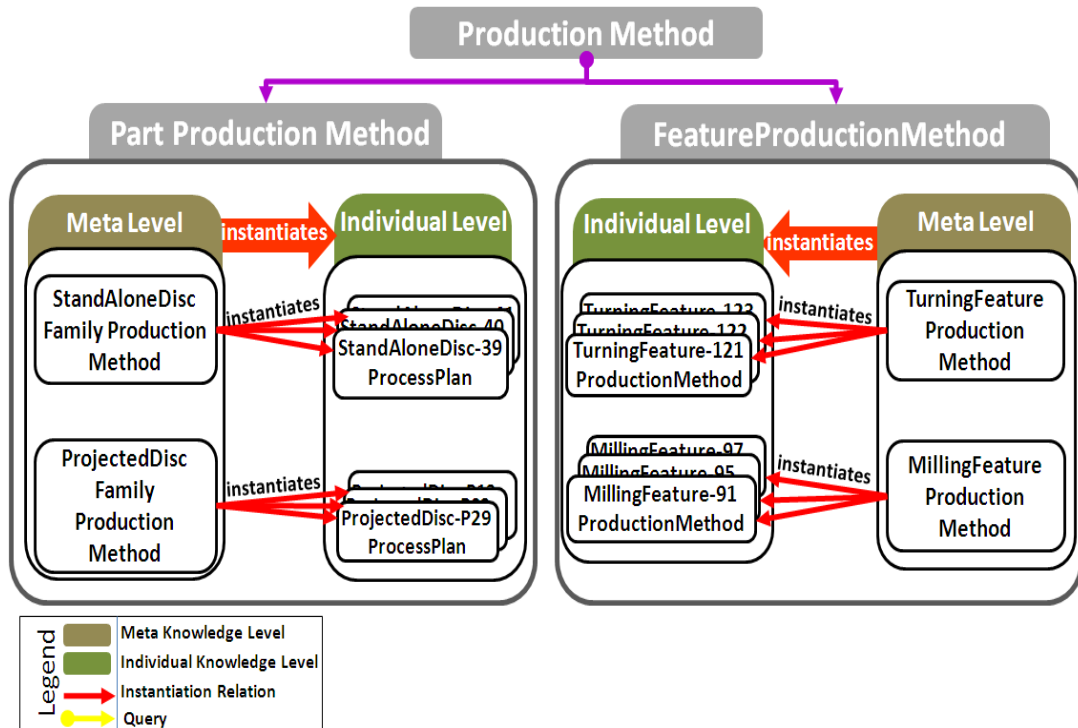


Figure 7.1: Requirements of capturing feature and Part Manufacturing Methods at Individual and Meta levels

Similarly, on the right hand side in figure 7.1, the *individual level* feature production methods are shown as being instantiated from their corresponding *Meta level* feature production methods.

The capture of these production methods at multiple levels of knowledge abstraction can enable production engineers to interact at these levels of knowledge and thus facilitate the early elimination of errors during production planning. This can assist in devising detailed individual process plans for a new or a modified part with fewer errors and thereby in reduced time. Production engineers have a need to reason about the accommodation of production methods for features within the existing production methods for parts. They may need to make queries at the *individual level knowledge* e.g. the following query is at individual level 'In which *ProcessPlan* can TurningFeature-121ProductionMethod be accommodated?'

This sort of queries can establish the production part families to which a feature can belong. However, it is understood from the industrial study that production engineers are interested in identifying the part families of production features early during production planning. They may need to make queries at the *Meta level knowledge*

e.g. The following query is at Meta level ‘In which *PartFamilyProductionMethod* can *TurningFeatureProductionMethod* be accommodated?’

The answer to this and other queries of this sort can establish the production part family to which a production feature can belong during the early phases of production. The concepts and relations to capture and query *individual level* production methods have already been reported in section 5.2.5 of chapter 5. However, the concepts and relations to capture and reason about *Meta level* production methods need to be explored.

7.3 Knowledge Capture and Reasoning at Meta Level

The concepts and relations defined in section 5.2.5 of chapter 5 provide the necessary basic understanding required to define the concepts and relation for *Meta level knowledge*. This section explains how the production method can be captured at the *Meta level knowledge* including the sequencing of events involved in production methods. The section also presents the formalisation of structures for reasoning about the manufacturability of the production methods of features in the production methods of part families.

7.3.1 Capturing Meta level Feature and PartFamily Production Methods

7.3.1.1 The Need to Define the *MetaMeta level* Structure for the *Meta level Knowledge*

In order to reason about the *Meta level knowledge*, the *Meta level knowledge* should be instantiated from an underlying structure. This means that the concepts and relations that instantiate the *Meta level knowledge* are required. Those concepts and relations will, therefore, be defined at an even higher level of abstraction which can be termed as the *MetaMeta level*. This follows from Gonzalez-Perez, et al (2004) who have mentioned that the concepts are present at the Meta level. This implies that the concepts from which the *Meta level knowledge* is instantiated are at the *MetaMeta level*. This shows that in order to reason about the *Meta level knowledge*, a *MetaMeta level* structure is required. The *MetaMeta level* enables the system to treat the *Meta level knowledge* as an instance which makes it possible to capture and reason about the *Meta level knowledge*.

This is illustrated figure 7.2 where the right hand side consists of *individual level knowledge* which is instantiated from the *Meta level knowledge*. The *Meta level knowledge* is at the centre of figure 7.2. The *individual level knowledge* e.g. *StandAloneDiscSa-123ProcessPlan* and *ProjectedDisc-P29ProcessPlan* can be reasoned about through their respective underlying Meta level structures i.e.

StandaloneDiscFamilyProcessPlan and *ProjectedDiscFamilyProcessPlan*. However, reasoning about these *Meta level structures* requires an underlying *MetaMeta level structure*. The figure shows an example of that *MetaMeta level structure* on the extreme left hand side. The *MetaMeta level structure* instantiates the *Meta level knowledge* and provides a reference for reasoning about the *Meta level knowledge*. This highlights the need to define the *MetaMeta level concepts* and relations.

7.3.1.2 The Use of ‘Clabjects’ and ‘Powertypes’

The work done on capturing multiple levels of concepts (Atkinson and Kühne 2000; Henderson-Sellers 2011; Henderson-Sellers and Gonzalez-Perez 2005; Henderson-Sellers and Hawryszkiewicz 2008; Palmer et al. 2011) provides help in defining the *MetaMeta level structure*. A concept lies at *Meta level* because it provides the structure that instantiates individuals. However, in this thesis concepts are to be treated as instances. Therefore, a new term is required to refer to such concepts that are instances of other concepts. The concept of ‘clabject’ introduced by Atkinson and Kühne (2000) is helpful in this regard. A clabject is used to model the concepts that

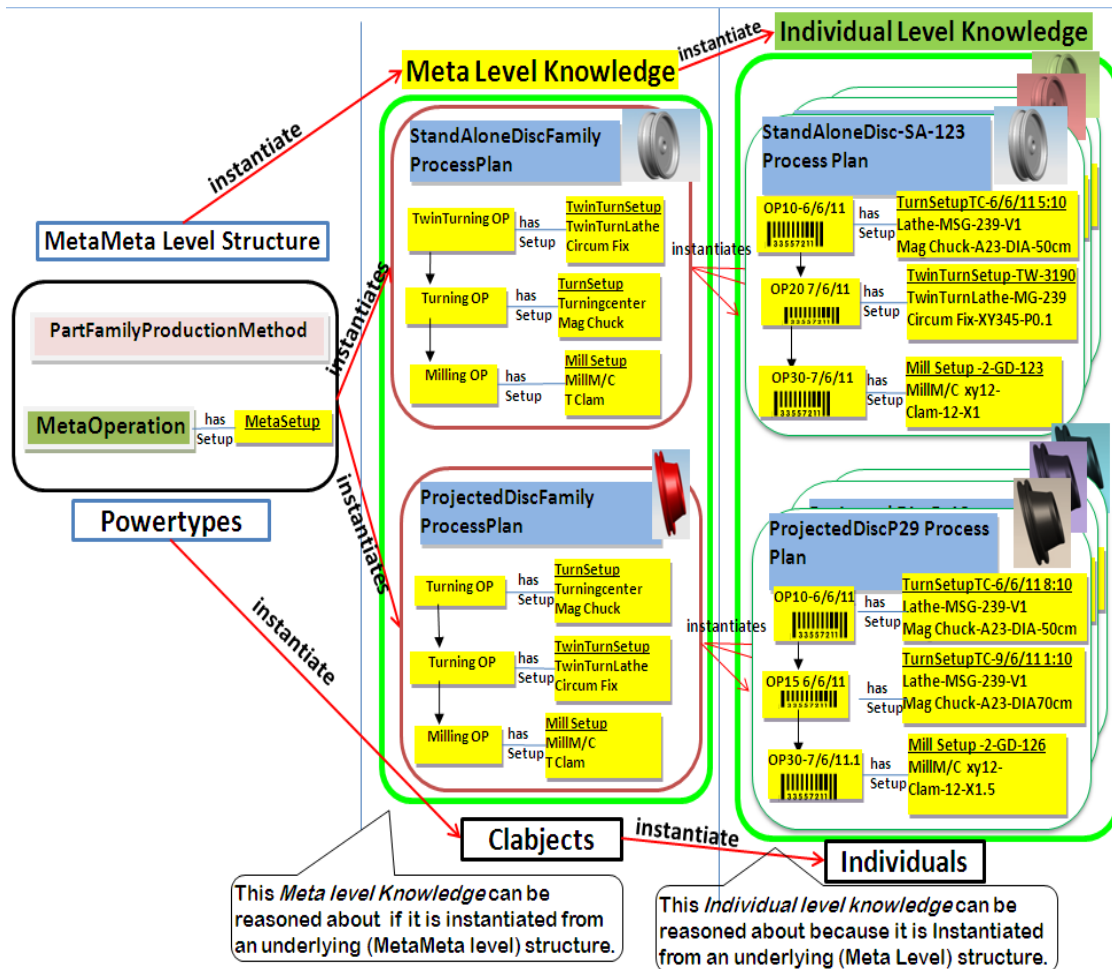


Figure 7.2: Manufacturing methods for part at individual and Meta levels

are instances of other concepts (Henderson-Sellers 2011) and this, therefore, poses a need to define the concepts that instantiate clabjects.

The concepts that instantiate clabjects are called ‘powertypes’ (Gonzalez-Perez and Henderson-Sellers 2006; Henderson-Sellers and Gonzalez-Perez 2005). The instances of powertypes i.e. clabjects can be sub-concepts of other concepts but they cannot be individuals.

Using powertypes and clabjects, the *Meta level knowledge* can be captured and reasoned about because they provide a means of treating the concepts as instances. Consequently, it is possible to support reasoning ability over clabjects. Because clabjects are at the Meta level, the knowledge associated with clabjects is *Meta level knowledge*. The formal logic captured using powertypes can provide an underlying structure for the *Meta level knowledge*. Figure 7.2 illustrates the approach by providing examples of powertypes, clabjects and individuals for part production method. The example of required powertypes (on the extreme left hand side) that instantiate the *Meta level* process plans (in the centre) are given in figure 7.2. In that example, the concept *PartFamilyProductionMethod*, is a powertype that instantiates the clabjects *StandAloneDiscProcessPlan* and *ProjectedDiscProcessPlan*. At the same time, the clabjects *StandAloneDiscProcessPlan* and *ProjectedDiscProcessPlan* can be sub concepts of the concept *ProcessPlan*. This is illustrated in figure 7.3.

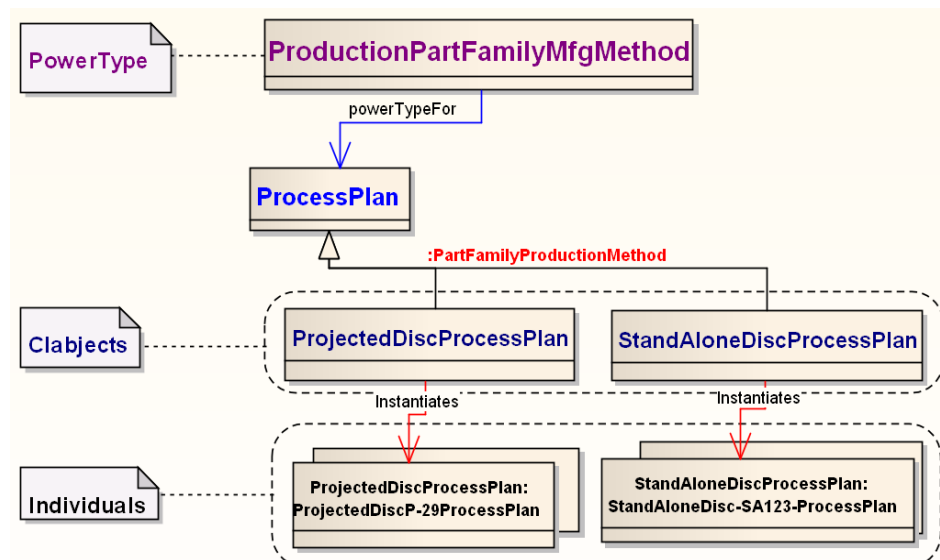


Figure 7.3: Lightweight formalisation of *ProcessPlan*'s powertype, clabjects and their individuals

As shown in figure 7.3, '*PartFamilyProductionMethod*' is a powertype for *ProcessPlan*. This means that the sub-concepts of *ProcessPlan* i.e. clabjects '*StandAloneDiscProcessPlan*' and '*ProjectedDiscProcessPlan*' are instances of *PartFamilyProductionMethod*. The standard way of representing in UML-2 that sub-

concepts of a concept are clabjects of a powertype is to put a colon symbol “:” followed by the name of the powertypes on the inheritance relation as shown by the red coloured “:PartFamilyProductionMethod” in figure 7.3. Each clabject itself can be instantiated several times to capture the individuals as shown in the lower half of figure 7.3. The KFL declaration of concepts in figure 7.3 is given below. The term used to refer to powertypes in KFL is ‘MetaProperty’. The KFL declaration of powertype *PartFamilyProductionMethod* is as follows.

```
:Prop      PartFamilyProductionMethod
:Inst      MetaProperty
:sup       Type
:metaPropFor ProcessPlan
```

The clabjects *StandAloneDiscProcessPlan* and *ProjectedDiscProcessPlan* are declared as follows:

```
:Prop StandAloneDiscProcessPlan      :Prop ProjectedDiscProcessPlan
:Inst PartFamilyProductionMethod      :Inst PartFamilyProductionMethod
:sup  ProcessPlan                     :sup  ProcessPlan
```

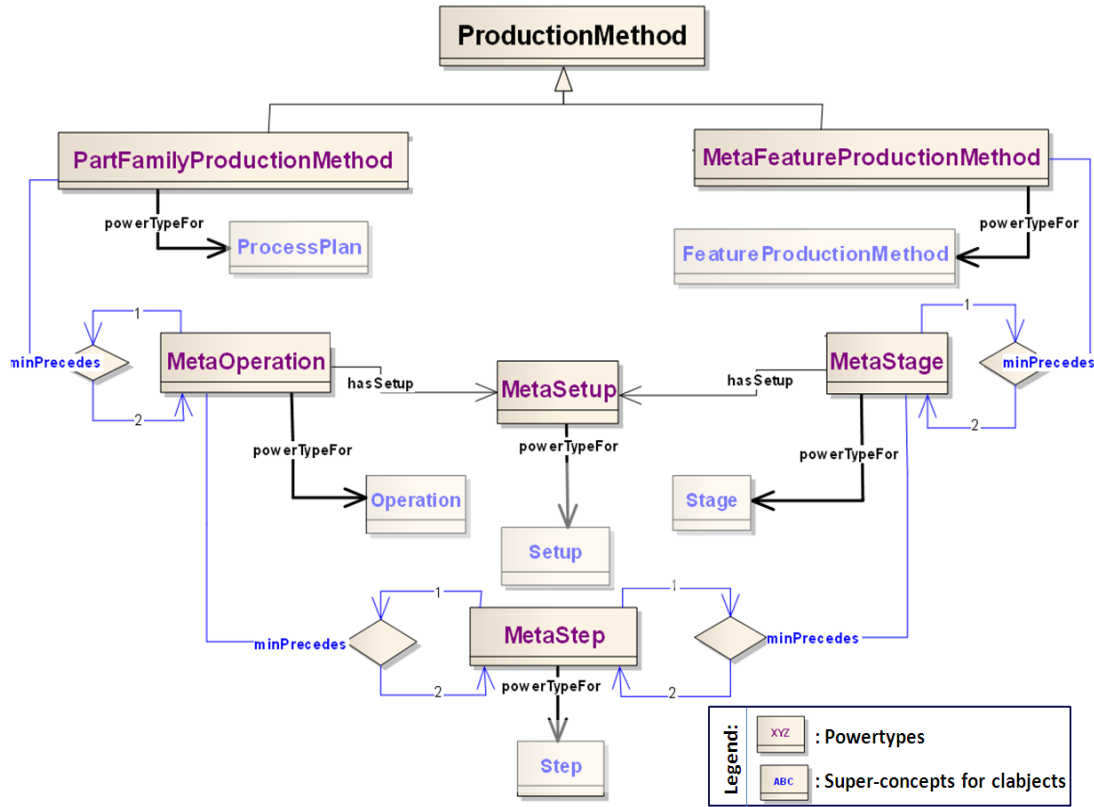
The above declaration shows that the clabjects *StandAloneDiscProcessPlan* and *ProjectedDiscProcessPlan* are instances of powertype *PartFamilyProductionMethod* and sub-concepts of the *ProcessPlan*. The concept that becomes part of the MCCO is the powertype *PartFamilyProductionMethod* whereas its clabjects *StandAloneDiscProcessPlan* and *ProjectedDiscProcessPlan* are only examples. Similarly, the powertype for reasoning the *FeatureProductionMethod* clabjects is *MetaFeatureProductionMethod*.

Following the same approach, the powertypes for the clabjects of *Operation*, *Stage*, *Setup* and *Step* have been identified as *MetaOperation*, *MetaStage*, *MetaSetup* and *MetaStep*.

These powertypes are used to extend the production method structure presented in figure 5.10 in chapter 5 to be applicable to the Meta level production methods. The extended structure is shown in figure 7.4, where the ‘powertypes’ are introduced to make the formalisation applicable to *Meta level knowledge* as well.

Like figure 5.10, figure 7.4 shows the *ProductionMethod* for part families on the left and the *ProductionMethod(s)* for features on the right hand side. The use of the powertypes, as shown by the highlighted concepts in figure 7.4, makes the structure applicable to Meta Level production methods. The use of the relation ‘*powerTypeFor*’ shown in figure 7.4 has already been explained with example in figure 7.3. In figure 7.4 the relations *minPrecedes* and *hasSetup* have only been defined over powertypes and not other concepts. This is because the relations defined over powertypes will be

applicable to their instantiated clabjects. This is why the relations defined between concepts in figure 5.10 have been replaced by the relations between powertypes in figure 7.4. Figure 7.4 presents the lightweight formalisation applicable to both *individual* and *Meta level* production methods.



7.4: Structure for the acquisition of meta level manufacturing method

7.3.2 Capturing Sequencing of the Relation ‘minPrecedes’

The capture of sequencing requires a relation that not only captures the sequencing semantics for *Operations*, *Stages*, and *Steps*, but also relates them to their corresponding *ProcessPlan* and/or *FeatureProductionMethod*. Moreover, this relation is required for both the levels of knowledge abstractions.

As mentioned in section 5.2.5 of chapter 5, the relation *minPrecedes* from PSL (ISO-18629-1 2004) has been used for this purpose. However, the required sequencing semantics cannot be satisfied by the PSL semantics of *minPrecedes*. In PSL, terms *Activity* and *ActivityOccurrence* respectively represent the events and their occurrences in a process. For example, ‘Paint house’ is an *Activity* and ‘Paint house at 12:30’ is an *ActivityOccurrence*.

In the context of this thesis, *Activities* and *ActivityOccurrences* can be considered equivalent to concepts and individuals respectively. According to its PSL semantics, *minPrecedes* can only capture the sequencing of *ActivityOccurrences*, which implies

that *minPrecedes* can only be used for *individual level knowledge*. However, in this thesis, there is a requirement to capture and reason about the *Meta level knowledge*.

In order to solve this problem the semantics of relation *minPrecedes* are tailored to make it applicable to multiple levels of knowledge abstraction.

7.3.2.1 Lightweight Formalisation of ‘minPrecedes’

The lightweight representation of the tailored *minPrecedes* is shown in figure 7.5.

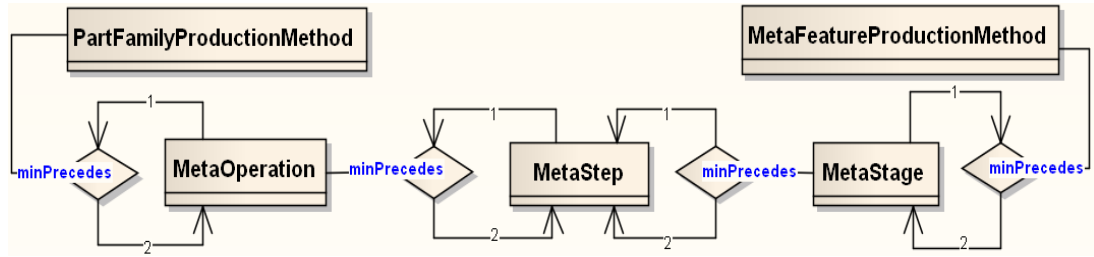


Figure 7.5: Lightweight formalisation of relation 'minPrecedes'

Relation *minPrecedes* is formalised to be applicable to the following conditions:

1. In a *PartFamilyProductionMethod* a *MetaOperation* *minPrecedes* another *MetaOperation*
2. In a *MetaFeatureProductionMethod* a *MetaStage* *minPrecedes* another *MetaStage*
3. In a *MetaStage*, a *MetaStep* *minPrecedes* another *MetaStep*
4. In a *MetaOperation*, a *MetaStep* *minPrecedes* another *MetaStep*

Figure 7.5 presents the formalisation based on powertypes that makes *minPrecedes* applicable to clajects instantiated from the powertypes. The structure in figure 7.5 is also valid for individual level production knowledge *because* a relation that is applicable to clajects is also applicable to the individuals instantiated from those clajects. The utilization of *minPrecedes* is further illustrated with the help of examples.

Consider a *FeatureProductionMethod* ‘*CircumferentialGrooveProductionMethod*’ having a sequence of *Stages*: ‘*RoughTurningStage* followed by *GroovingStage* followed by *FinishTurningStage*’, this sequence can be captured using *minPrecedes* as shown in figure 7.6a. Similarly, the sequence of *Operations* for *StandAloneDiscFamilyProductionMethod* can be captured as shown in Figure 7.6b using *minPrecedes*. In a similar manner, *minPrecedes* can be applied to capture the sequencing of *Steps* in a *Stage* or an *Operation*.

The diamond shape in UML can be used to capture the ternary relations (Palmer et al. 2011).

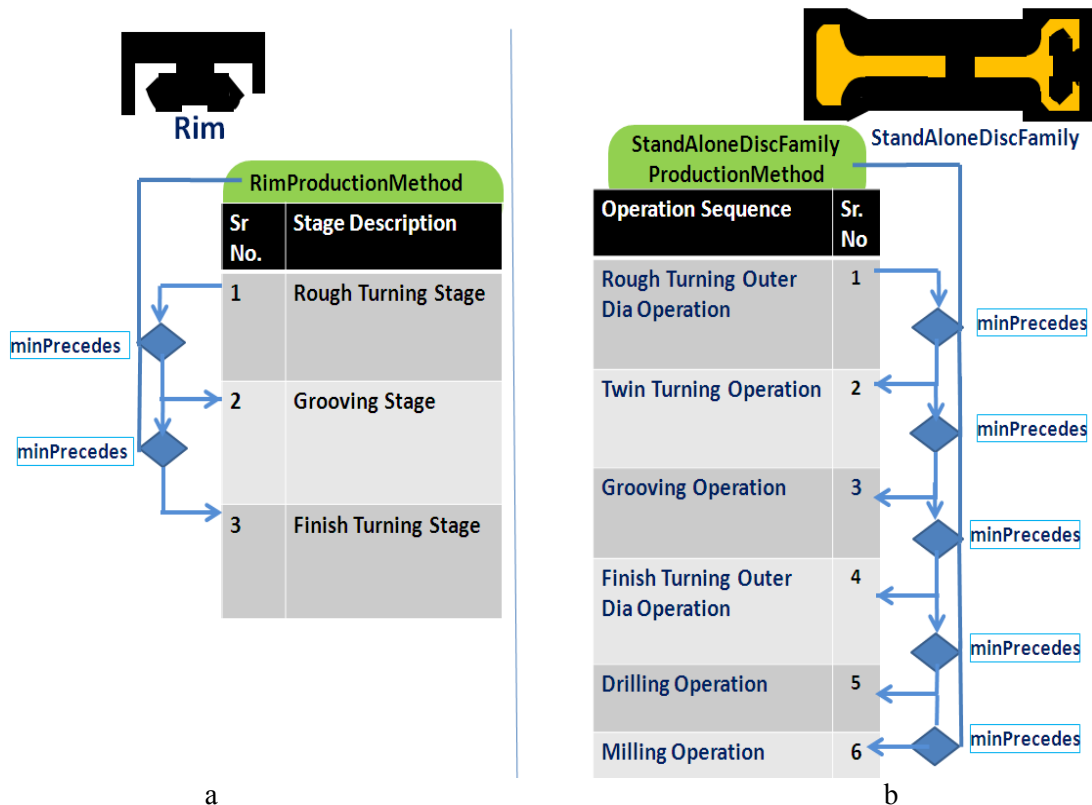


Figure 7.6: a) Relation *minPrecedes* to capture sequencing of stages in a *RimProductionMethod*,
b) Relation *minPrecedes* to capture sequencing of Operation in a *StandAloneDiscFamilyProductionMethod*

7.3.2.2 Heavyweight Formalisation of ‘minPrecedes’

The relation *minPrecedes* is formalised in way which is applicable for capturing the sequencing at individual *level knowledge* as well as *Meta level knowledge*.

```
:Rel minPrecedes
:Inst TernaryRel
:Sig Top Top Top
```

As shown, *minPrecedes* is declared as a ternary relation. The directive ‘:Sig’ specifies all three arguments of *minPrecedes* to be ‘Top’. ‘Top’ is a foundation concept referenced from ULO. The use of *Top* makes the relation applicable to individuals, clabjects, and powertypes. This in other words makes the relation *minPrecedes* applicable to both levels of knowledge abstraction. But, having the concept *Top* as the arguments of *minPrecedes* means that *minPrecedes* can be applicable on any concept and its instances from anywhere in the ontology. However, it is understood that *minPrecedes*, for *Meta level knowledge*, should be applicable to the following:

1. Clabjects instantiated from *MetaOperation* in a clabject of *PartFamilyProductionMethod*
2. Clabjects instantiated from *MetaStage* in a clabject of *MetaFeatureProductionMethod*
3. Clabjects instantiated from *MetaStep* in a clabject of *MetaOperation* or *MetaStage*

This range of applicability is formally captured through the following axioms.

;IC defining the range of applicability of *minPrecedes* for Meta level knowledge

```
(=> (minPrecedes ?x ?y ?mm)
  (or (and (MetaOperation ?x)
            (MetaOperation ?y)
            (PartFamilyProductionMethod ?mm))
      (and (MetaStage ?x)
            (MetaStage ?y)
            (MetaFeatureProductionMethod ?mm))
      (and (MetaStep ?x)
            (MetaStep ?y)
            (MetaStage ?mm))
      (and (MetaStep ?x)
            (MetaStep ?y)
            (MetaOperation ?mm))))
```

:IC hard "The relation *minPrecedes* holds between clabjects instantiated from *MetaStages* for corresponding clabject of *MetaFeatureProductionMethod* or between clabjects of *MetaOperation* and their corresponding clabject of *PartFamilyProductionMethod* or between clabjects of *MetaStep* and their corresponding clabject of *MetaStage* or *MetaOperation*"

Similarly, it is also understood that *minPrecedes*, for *individual level knowledge*, should be applicable to the following:

1. Individuals instantiated from *Operation* in an individual of *ProcessPlan*
2. Individuals instantiated from *MetaStage* in an individual of *FeatureProductionMethod*
3. Individuals instantiated from *Meta Step* in an individual of *Operation* or *Stage*

An IC similar to the last IC formally captures this range of applicability as well. After the capture of the range of applicability, the linear ordering semantics also need to be captured. Linear ordering semantics ensure that a sequence of occurrence of events in a production method is correct. The linear ordering semantics specifically ensure the following properties of *minPrecedes* in a single sequence, (1.) Irreflexivity: Same event does not occur twice, (2) Antisymmetry: An event which is occurring later does not occur earlier, (3) Transitivity. If an event 'a' precedes an event 'b' and 'b' precedes 'c', then 'a' precedes 'c'. The formalisation of these properties is given below.

```
(=> (and (Top ?arg1)
         (Top ?arg2)
         (Top ?arg3)
         (minPrecedes ?arg1 ?arg2 ?arg3))
  (not (= ?arg1 ?arg2)))
:IC hard "minPrecedes is Irreflexive."
```

This IC ensures that same item does not occur twice in a single *ProductionMethod*

```
(=> (and (Top ?arg1)
         (Top ?arg2)
         (Top ?arg3)
         (minPrecedes ?arg1 ?arg2 ?arg3))
  (not (minPrecedes ?arg2 ?arg1 ?arg3)))
:IC hard "minPrecedes is Antisymmetric."
```

This IC ensures that an item which is occurring later does not occur earlier in a sequence

```
(=> (and (Top ?arg1)
         (Top ?arg2)
         (Top ?arg3)
         (Top ?arg4)
         (minPrecedes ?arg1 ?arg2 ?arg3)
         (minPrecedes ?arg2 ?arg4 ?arg3))
      :rem "minPrecedes is Transitive."
```

This inference rule ensures that *minPrecedes* is Transitive.

It is to be recalled (from section 7.2) that one of the requirements is to reason about the accommodation of production methods for features within the production methods for part families and the next section focuses on this

7.3.3 Reasoning about Feature and PartFamily Production Methods

The understanding gained from the definition of production methods for feature and parts provides a basis for reasoning about them. The required reasoning logic can be stated as:

“A *FeatureProductionMethod* can be accommodated within a *PartFamilyProductionMethod* if all the *Stages* of a *FeatureProductionMethod* can be manufactured with correct sequencing in the *Operations* of a *PartFamilyProductionMethod*.”

This logic is quite complex to formalise particularly because

1. It is necessary to check not only the manufacturability of stages (of feature production methods) in operations (of part family production methods) but, also to ensure their correct sequencing.
2. The logic has to be applicable at Meta level knowledge.

The process of formalising this logic is divided into two steps. First, the logic to find out the manufacturability of *Stages* (of a *FeatureProductionMethod*) within the *Operations* (of the *PartFamilyProductionMethod*) irrespective of their sequencing is formalised. Second, the logic to handle the issue of correct sequencing is formalised.

7.3.3.1 Reasoning About the Manufacturability of Stages in Operations

In order to formalise a method of finding out the manufacturability of the *Stages* in *Operations*, a relation *canBeManufacturedIn* is defined. For the relation *canBeManufacturedIn* to work, the system should understand the condition under which a *Stage* can be manufactured in an *Operation*.

From the definitions of *Stage* and *Operation*, it is known that both have unique Setups. For a *Stage* to be manufacturable within an *Operation* its *Setup* should either be the same or similar to the *Setup* of the *Operation*. This is illustrated with help of an example in figure 7.7.

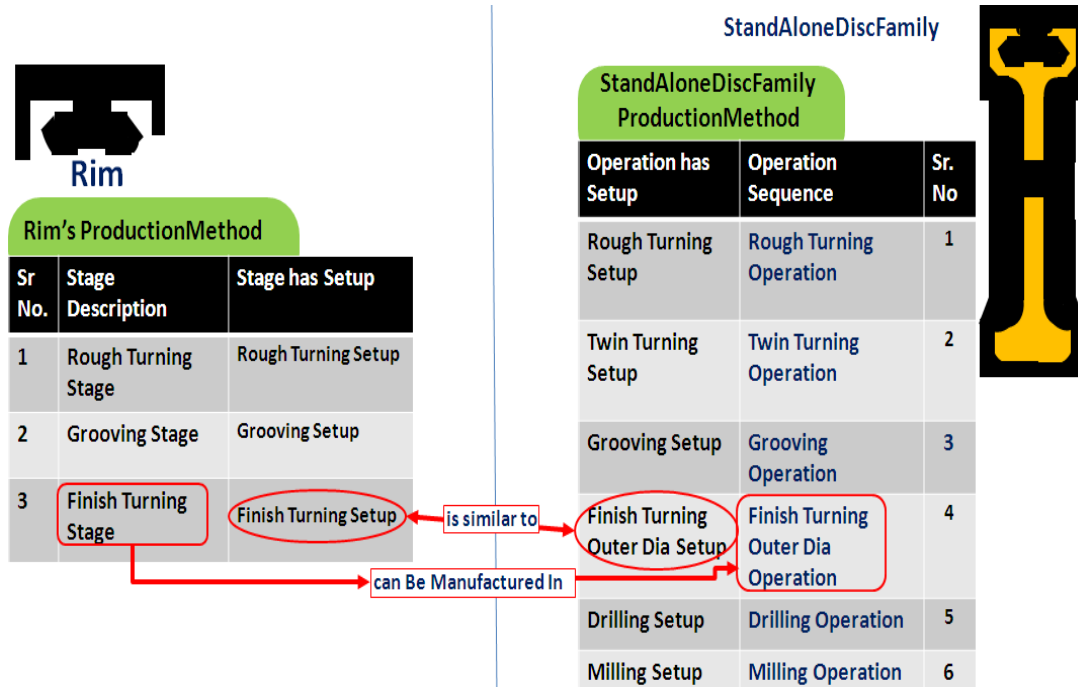
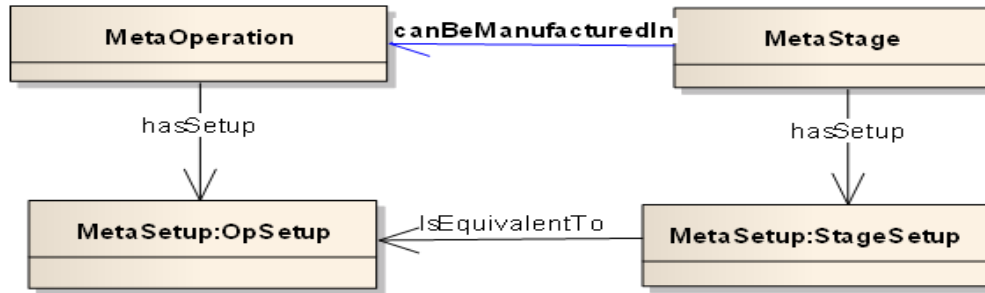


Figure 7.7: Illustration of the relation *canBeManufacturedIn*

On the right hand side in Figure 7.7, a *PartFamilyProductionMethod* namely *StandAloneDiscFamilyProductionMethod* is shown with its sequence of *Operations*. Each *Operation* has a unique *Setup* with respect to the other *Operations*. Similarly on the left hand side in figure 7.7, a *MetaFeatureProductionMethod* namely *RimProductionMethod* is shown with its sequence of *Stages*. Each *Stage* has a unique *Setup* with respect to the other *Stages*.

As mentioned earlier, for a *Stage* to be manufactured in an *operation*, *Setup* of the *Stage* should be same or similar to that of the *Operation*. As shown in the figure 7.7, the *Setup* of the *FinishTurningStage* is similar to the *Setup* of the *FinishTurnigOuterDiaOperation*. Therefore, the *FinishTurningStage* *canBeManufacturedIn* the *FinishTurningOuterDiaOperation*. This is also represented in figure 7.7. Similarly, other *Stages* of *RimProductionMethod* *canBeManufacturedIn* the *Operations* whose *Setups* are same or similar to the *Setups* of *Stages*.

Figure 7.8 presents a lightweight formalisation of the semantics of relation *canBeManufacturedIn*.



7.8: Lightweight formalisation of relation 'canBeManufacturedIn'

The KFL formalisation of *canBeManufacturedIn* is given below.

```
:Rel canBeManufacturedIn
:Inst QuaternaryRel
:Sig Top Top Top Top
```

The relation is declared as a quaternary relation because it requires four arguments. These arguments are (1) *Stage*, (2) *FeatureProductionMethod* (to which *Stage* belongs), (3) *Operation*, and (4) *ProcessPlan* (to which that *Operation* belongs). Since the relation is to be applicable at both *individual* and *Meta level knowledge* therefore, the arguments of the relation are *Top*. The range of applicability is defined through the following axioms.

```
;ICs to formally define the range of applicability and the four arguments of
canBeManufacturedIn
(=> (canBeManufacturedIn ?a ?b ?x ?y)
  (or (and (MetaStage ?a)
    (MetaFeatureProductionMethod ?b)
    (MetaOperation ?x)
    (PartFamilyProductionMethod ?y))))))
:IC hard "The relation canBeManufacturedIn holds between instances of MetaStage
belonging to an instance of MetaFeatureProductionMethod, clobjects of MetaOperation
belonging to a clobject of PartFamilyProductionMethod."
```

A similar IC is written to make the relation applicable to *Individual level knowledge*.

The next axiom, for *Meta level knowledge*, captures the semantic of *canBeManufacturedIn*.

```
;Inference rule applicable at level Meta knowledge to establish the Stage type that
canBeManufacturedIn an Operation type
(<= (canBeManufacturedIn ?st ?mfpm ?op ?pfpm)
  (and (MetaFeatureProductionMethod ?mfpm)
    (MetaStage ?st)
    (MetaSetup ?set1)
    (hasStage ?fpm ?st)
    (hasSetup ?st ?set1)
    (PartFamilyProductionMethod ?pfpm)
    (MetaOperation ?op)
    (MetaSetup ?set2)
    (hasSetup ?op ?set2))
```

```
(= ?set1 ?set2)))
:rem "The clbject of MetaStage ?st belonging to the clbject of
MetaFeatureProductionMethod ?mfpm can be manufactured within the clbject of
MetaOperation ?op belonging to the clbject of PartFamilyProductionMethod ?pfpm, since
?st and ?op share the same clbject of MetaSetup."
```

A similar axiom is also defined for *individual level knowledge*. The above mentioned axioms make it possible to query the stages of feature production methods that can be manufactured in the operations of part family production methods. However, this is not enough to query the manufacturability of a *FeatureProductionMethod* in a *ProcessPlan*. It is possible that even if all the *Stages* of a *FeatureProductionMethod* *canBeManufacturedIn* the *Operations* of a *ProcessPlan*, and still, the *FeatureProductionMethod* cannot be accommodated in the *ProcessPlan*. This is because the relation *canBeManufacturedIn* does not deal with sequencing.

7.3.3.2 Logical Reasoning to Ensure Correct Sequencing

In order to find out whether a *FeatureProductionMethod* can be accommodated in a *ProcessPlan*, a relation '*canBeAccommodatedIn*' is identified which can be stated as;

- *FeatureProductionMethod canBeAccommodatedIn PartFamilyProductionMethod*

For the relation *canBeAccommodatedIn* to hold true, the system has to understand and reason the *Stage* sequence of the *FeatureProductionMethod* against the *Operation* sequence of the *ProcessPlan*. This is a complex issue which is explained with the help of example cases shown in figure 7.9.

In the two diagrams in figure 7.9, *RimFeatureProductionMethod* and its sequence of *Stages* is shown on the left hand side. On the right hand in diagram showing 'Case A', a *ProjectedDiscFamilyProductionMethod* is shown with its sequence of *Operations*. On the right hand side in diagram showing 'Case B' a *StandAloneDiscFamilyProductionMethod* is shown with its sequence of *Operations*. In both i.e. Case A and B, the different *Stages* of *RimProductionMethod* can be manufactured in their corresponding *Operations* as shown in figure 7.9.

Case 'A' present the 'False' and case 'B' presents the 'True' condition under which a *FeatureProductionMethod* can be accommodated in a *PartFamilyProductionMethod*.

As an illustration of the false condition, consider Case 'A'. In case A it is shown that *Stage 1* of *RimProductionMethod* is manufactured after *Stage 2* because the *Operation* corresponding to *Stage 2* i.e. *Operation 1* occurs first. *Stage 1* can be manufactured after *Stage 2* because the *Operation* in which *Stage 1* can be manufactured i.e. *Operation 4* occurs at 4th place (after *Operation 1*) in the *ProjectedDiscProductionMethod*.

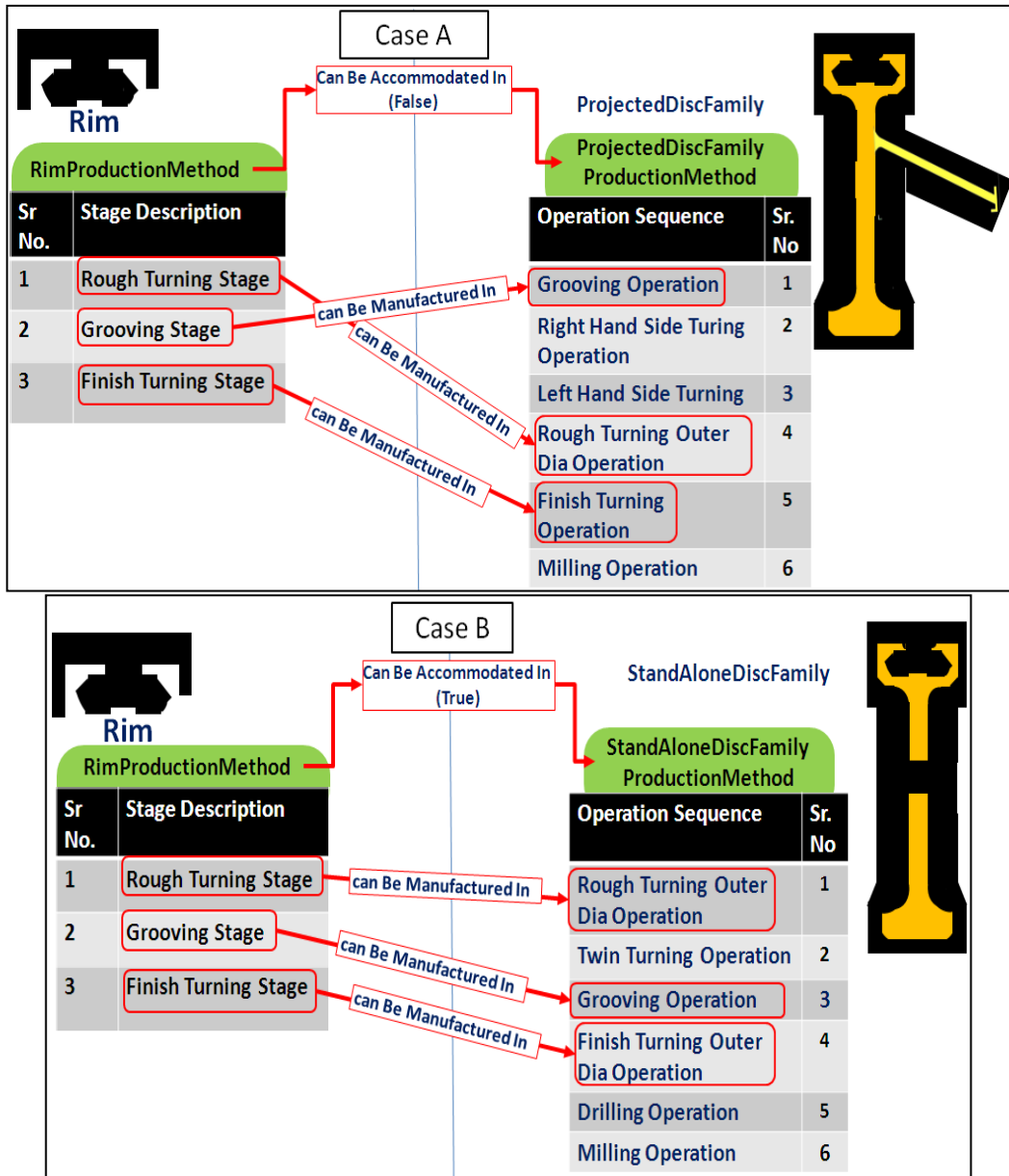


Figure 7.9: True and False conditions for FeatureProductionMethod to be accommodated in a PartFamilyProductionMethod

It is shown that the *RimProductionMethod* can be accommodated in *ProjectedDiscProductionMethod*. However, this is impossible because of the issues in sequence of manufacturing of Stages. Case A shows that *Stage2* i.e. GroovingStage of *RimProductionMethod* is manufactured before the *Stage 1*. However, in practice this is impossible to manufacture a *Stage* before its preceding *Stage* has been manufactured. Therefore, it is impossible to accommodate the *RimProductionMethod* in *ProjectedDiscProductionMethod*. Thereby, case A is false. This is why Case A has been marked as false. Example of case A shows that even if all the *Stages* of a *FeatureProductionMethod* can independently be manufactured within the *Operations* of a *PartFamilyProductionMethod*, the *FeatureProductionMethod* cannot be accommodated in the *PartFamilyProductionMethods* if the sequencing is not correct.

A *FeatureProductionMethod* can be accommodated in a *PartFamilyProductionMethod* if all the *Stages* of the *FeatureProductionMethod* are manufacturable with correct sequence in the *Operations* of the *PartFamilyProductionMethod* as shown by Case B in figure 7.9. Case B shows that the sequence of manufacturing of *Stages* of *RimProductionMethod* is maintained when they are manufactured in the *Operations* of the *StandAloneDiscProductionMethod*. Therefore, *RimProductionMethod* can be accommodated in *StandAloneDiscProductionMethod*.

In order for a *FeatureProductionMethod* to be accommodated within a *PartFamilyProductionMethod* the false condition should not hold true. Once the false condition holds false, then the remaining scenarios can only mean that the *FeatureProductionMethod* *canBeAccommodatedIn* *PartFamilyProductionMethod*. These semantics illustrated by these true and false conditions are captured formally in the formalisation of the relation *canBeAccommodatedIn*.

7.3.3.2.1 Formalising the Semantics of ‘canBeAccommodatedIn’

The first step is to declare the relation in KFL.

```
:Rel canBeAccommodatedIn
:Inst BinaryRel
:Sig Top Top
```

The relation has two arguments i.e. *FeatureProductionMethod* and *ProcessPlan*. However, this relation is required to be applicable at multiple levels of knowledge abstraction. Therefore, the arguments of this relation are declared as *Top*. The applicability range of this relation is then defined through the following axiom.

```
;IC to define the range of concepts on which canBeAccommodatedIn is applicable
(=> (canBeAccommodatedIn ?x ?y)
    (or (and (MetaFeatureProductionMethod ?x)
              (PartFamilyProductionMethod ?y))
        (and (FeatureProductionMethod ?x)
              (ProcessPlan ?y))))
:IC hard "The relation canBeAccommodatedIn holds between concepts of
FeatureProductionMethod and their instances and concepts of ProcessPlan and their
instances."
```

The next step is to capture the True and False condition for *canBeAccommodatedIn*. First an axiom is defined to capture the false condition. In order for the relation *canBeAccommodatedIn* to hold true, the rule which captures the false condition should be negated. First the axiom ‘*canBeAccommodatedIn_false*’ is defined to capture the false condition illustrated in case ‘A’ in figure 7.9 where, a later *Stage* is being manufactured before its preceding *Stage*. The KFL formalisation of the *canBeAccommodatedIn_false* is as follows.

;Inference rule applicable at Meta knowledge level: to infer a *FeatureProductionMethod* that *canBeAccommodatedIn* a *PartFamilyProductionMethod* based on a wrong condition i.e. a later stage being manufactured earlier than in its preceding Stage in the Operations of a *PartFamilyProductionMethod*

```
(<= (canBeAccommodatedIn_false ?mfpm ?pfpm)
      (and (MetaStage ?st1)
            (MetaStage ?st2)
            (minPrecedes ?st1 ?st2 ?fpm)
            (MetaOperation ?op1)
            (MetaOperation ?op2)
            (minPrecedes ?op1 ?op2 ?pfpm)
            (canBeManufacturedIn ?st1 ?fpm ?op2 ?pfpm)
            (canBeManufacturedIn ?st2 ?fpm ?op1 ?pfpm))))
```

The above axiom states that a cl object 'mfpm' of *MetaFeatureProductionMethod* can be accommodated in a clobject 'pfpm' of *PartFamilyProductionMethod* even if a clobject '?st2' of *MetaStage* that occurs later than the clobject '?st1' can be manufactured in clobject '?op1' of *MetaOperation* earlier than ?st1. However, this is not possible in reality, therefore, the directive '_false' is placed at the end of the inference relation. This directive specifies that if the relation *canBeAccommodatedIn_false* holds true, it captures a false condition and *FeatureProductionMethod* cannot be accommodated in *PartFamilyProductionMethod*. The above axiom is applicable to the *Meta level knowledge* and similar axiom is also written to make the relation *canBeManufacturedIn_false* applicable to the *individual level knowledge*.

The above rule is based on a false condition. In order for a *FeatureProductionMethod* to be accommodated within a *PartFamilyProductionMethod* the Inference relation *canBeManufacturedIn_false* should hold false. Once this is false, then the remaining sequence will be correct. The following inference rule captures this. This rule formally states that a *FeatureProductionMethod* *canBeAccommodatedIn* a *PartFamilyProductionMethod* if; (1) all the stages of *FeatureProductionMethod* *canBeManufacturedIn* the Operations of *PartFamilyProductionMethod* and (2) *canBeAccommodatedIn_false* holds false.

```
(<= (canBeAccommodatedIn ?fpm ?pfpm)
      (and (MetaStage ?st)
            (MetaOperation ?op)
            (canBeManufacturedIn ?st ?fpm ?op ?pfpm)
            (not (canBeAccommodatedIn_false ?fpm ?pfpm))))
:rem "The FeatureProductionMethod ?fpm can be accommodated in the
PartFamilyProductionMethod ?pfpm"
```

The above axiom is applicable to the *Meta level knowledge* and a similar axiom is written to make the semantics of 'canBeManufacturedIn' applicable to the *individual level knowledge* as well.

The relations *canBeManufacturedIn* and *canBeAccommodatedIn* enable querying the manufacturability of *FeatureProductionMethod* Within a *PartFamilyProductionMethods*.

7.3.4 Complete Lightweight Representation of ProductionMethod

Figure 7.4 presented earlier in the chapter is adapted by including the relations *canBeManufacturedIn* and *canBeAccommodatedIn* as shown in figure 7.10. This was required to provide a complete representation of production method. Figure 7.10 illustrates the complete lightweight formalisation of the conceptualisation of *ProductionMethod* for *individual level knowledge* as well as *Meta level knowledge*. This figure presents the formalisation of the concept *MetaFeatureProductionMethod* and *FeatureProductionMethod* and on the right hand side and the formalisation of concept *PartFamilyProductionMethod* and *ProcessPlan* on the left hand side. The key

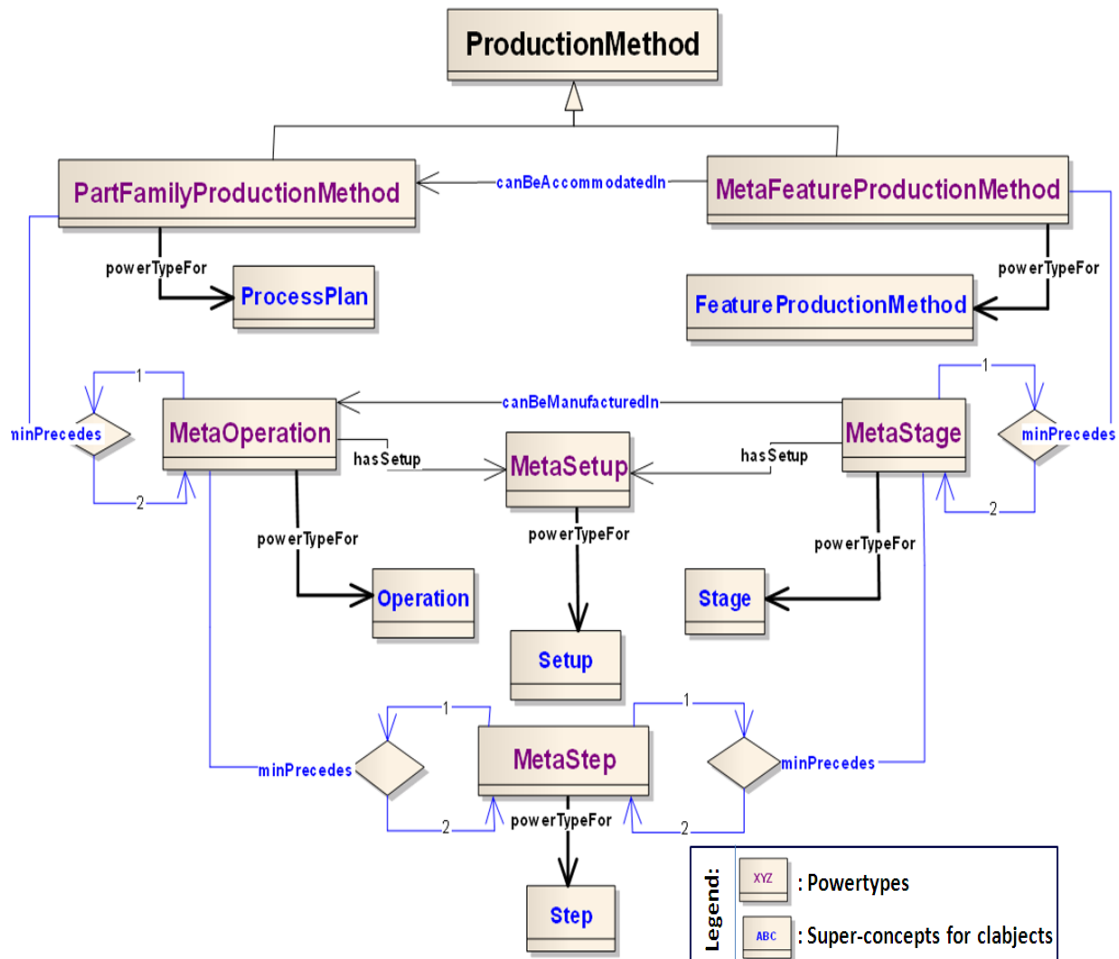


Figure 7.10: Complete lightweight formalisation of *ProductionMethod* as part of MCCO relations *canBeAccommodatedIn* and *canBeManufacturedIn* are also presented.

7.4 Summary

This chapter explicitly explains the requirements to capture and reason about the *Meta level knowledge*. The methodology and concepts have been identified to capture and reason about production methods for features and part families, particularly, at the *Meta level knowledge*. The proposed approach and set of concepts provide a significant step towards the understanding of the use of heavyweight ontologies to support capturing and reasoning about production methods at multiple levels of knowledge abstraction and particularly at *Meta level knowledge*. The methodology is also applicable to the *individual level knowledge*.

A comprehensive set of core manufacturing concepts i.e. *MetaOperation*, *Operation*, *MetaStage*, *Stage*, *MetaStep*, *Step*, *MetaSetup*, and *Setup* have been identified and formalised in KFL to capture production methods for features and part families at multiple levels of knowledge abstraction and particularly for *Meta level knowledge*. The relations *hasSetup* and *uses* help to formally define the concept of *MetaOperation*, *Operation*, *MetaStage*, *Stage*, *MetaStep*, *Step*, *MetaSetup*, and *Setup*.

The key relation *minPrecedes* has been defined to capture sequencing at both *Meta level knowledge* and *Individual level knowledge*. The highly complex axiomatization of the relations i.e. *canBeAccommodatedIn* and *canBeManufacturedIn*, enable reasoning about the manufacturability of production methods for features within the production methods for part families at *individual level knowledge* as well as *Meta level knowledge*. The next chapter experimentally investigates the solutions presented in this chapter along with the investigation of the solutions presented in chapters 5 and 6.

8 The Experimental verification of the research concept

8.1 Introduction

This chapter documents the design of the experimental system and the experiments to test different aspects of the research work reported in this thesis. The chapter provides an experimental proof of the research hypothesis and verifies the novel aspects of research work. The chapter is organized as follows. Section 8.2 illustrates the method of implementation and experimental verification of the ontology. Section 8.3 presents the implementation of the MCCO in the ontology development environment that captures the eight categories of concepts. Section 8.4 presents an overview of the experiments. Sections 8.5 to 8.8 report the experiments to verify the different research aspects reported in chapters 5, 6 and 7. Section 8.9 reports a case study to further strengthen the claim of research contribution and section 8.10 presents a summary of the chapter.

8.2 Method of Implementation and Experimental Verification

An illustration of the experimental setup is given in figure 8.1. This figure summarises

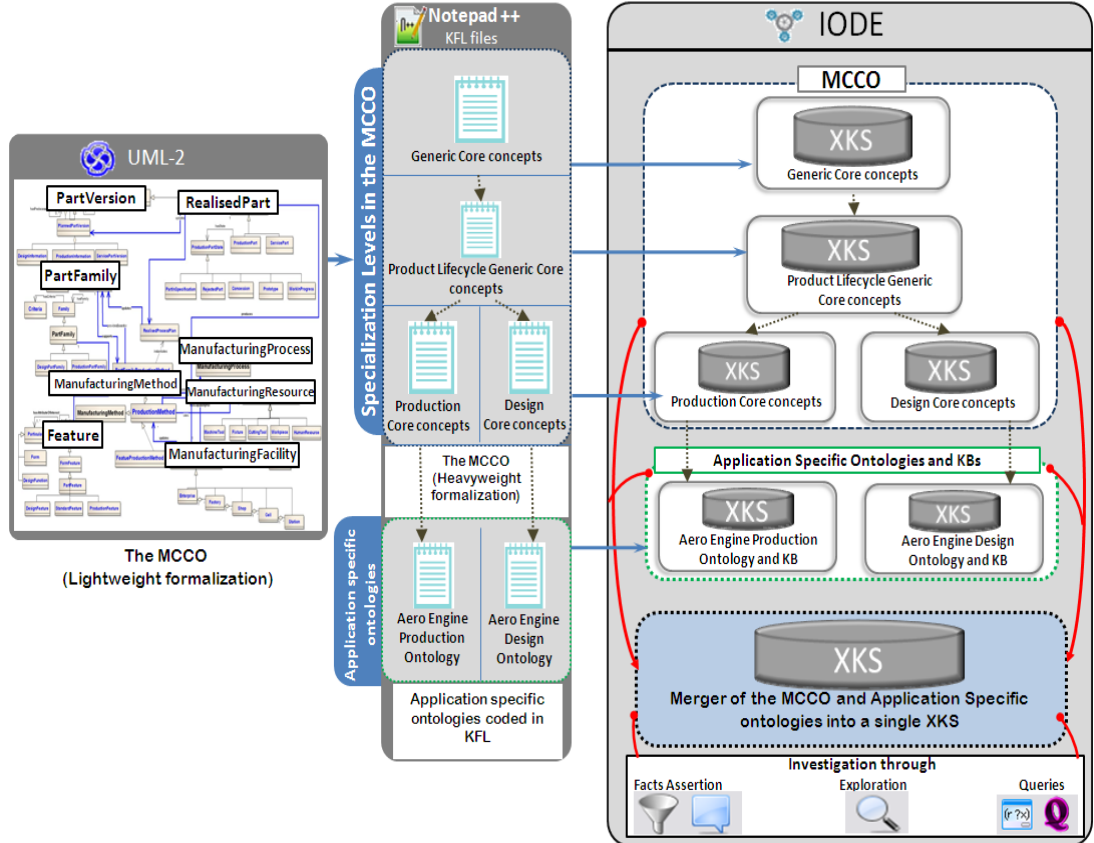


Figure 8.1: Scheme for the formalization of the MCCO and its experimental investigation

the procedure of experimental verification. The figure shows how the ontology is transformed from a lightweight model to a heavyweight one. The figure also shows the intention for the implementation of the ontology in the IODE and the use of IODE tools for the experimental verification. The figure is divided into three main portions. These portions, from left to right, are UML-2, Notepad++ and IODE.

The UML-2 portion represents the lightweight formalization of the ontology in UML-2, which has been presented in detail in chapter 5 as well as in chapters 6 and 7.

The Notepad++ portion shows how the ontology's KFL files have been coded at the three levels of specializations. The coding in KFL formalises the ontology in heavyweight logic. The Notepad++ portion also represents the coding of application specific design and production ontologies developed using the MCCO.

The IODE portion is the most important one with respect to the experimental verification. This portion shows the implementation of the ontology in the IODE. This portion shows that different levels of specialization of the MCCO are loaded in separate files in the IODE as different XKSs (figure 8.1). An XKS in IODE corresponds to a system that holds an ontology coded in KFL with a linked KB for asserting facts based on the ontology and making queries. The IODE portion also shows the implementation of the application specific design and production ontologies as different XKSs.

Then, it is shown that the MCCO and the application specific domain ontologies are merged into a single XKS. This is required to verify the specialization levels, development of specialised domain ontologies and knowledge sharing across them. This provides a basis for running suitable queries and processing results. Results can then be analysed in IODE itself or saved for other external transactions to facilitate the testing of the ontology and the different aspects of the research. As shown in the figure 8.1, different modules of the IODE i.e. Fact Assertor, XKS explorer and the query tools are used in the experimental verification.

8.3 Implementation of the MCCO in IODE

The first step in the experimental verification is to deploy the ontology into the IODE. A number of KFL files have been generated and loaded into IODE. The initial implementation of the MCCO in IODE is elaborated below.

Figure 8.2 presents an amalgamation of a number of screen shots to show the successful implementation of the MCCO. It shows the taxonomies of loaded concepts

([A] and [D]), view of their description [B], their relations [C] and the axioms formally defining the concepts [E]. The eight categories of the MCCO concepts are shown in

The figure displays the IODE browser interface for the MCCO ontology. It includes a tree view of concepts (A), a detailed description of a concept (B), a list of relations (C), a diagram of assertions (E), and a list of instances (F).

Panel A: MCCO - Property: Top - IODE Browser

- Search For: RootCtx.Top
- Subproperties of RootCtx.Top:
 - MCCO.RealisedPart
 - MCCO.ProductionPart
 - MCCO.ServicePart
 - MCCO.PartVersion
 - MCCO.PlannedPartVersion
 - MCCO.DesignPartVersion
 - MCCO.ProductionPartVersion
 - MCCO.ServicePartVersion
 - MCCO.ManufacturingResource
 - MCCO.CuttingTool
 - MCCO.Fixture
 - MCCO.HumanResource
 - MCCO.MachineTool
 - MCCO.Workpiece
 - MCCO.ManufacturingFacility
 - MCCO.Cell
 - MCCO.Enterprise
 - MCCO.Factory
 - MCCO.Shop
 - MCCO.Station
 - MCCO.ManufacturingMethod
 - MCCO.ProductionMethod
 - MCCO.FeatureProductionMethod
 - MCCO.Operation
 - MCCO.OperationSequence
 - MCCO.PartFamilyProductionMethod
 - MCCO.Setup
 - MCCO.Stage
 - MCCO.StageSequence
 - MCCO.Step
 - MCCO.StepSequence
 - MCCO.ManufacturingProcess
 - MCCO.Feature
 - MCCO.FormFeature
 - MCCO.PartFeature
 - MCCO.DesignFeature
 - MCCO.ProductionFeature
 - MCCO.StandardFeature
 - MCCO.Family
 - MCCO.PartFamily
 - MCCO.DesignPartFamily
 - MCCO.ProductionPartFamily

Panel B: Property: MCCO.Family

- General name: "Family"
- Instance of: RootCtx.Type
- Super-properties: MLO.Group
- Binding pattern: { }
- Remarks: A Family is a group defined on the basis of a common criteria

Panel C: Family has these super properties:

- MCCO.Family
 - MCCO.hasCriteria: MCCO.Family X MCCO.Criteria
 - MLO.Object
 - MLO.ConcreteEntity
 - RootCtx.Particular
 - RootCtx.Top

Panel D: MCCO.Family

```

graph TD
    MCCO.Family --> MCCO.PartFamily
    MCCO.PartFamily --> MCCO.ProductionPartFamily
    MCCO.PartFamily --> MCCO.DesignPartFamily
  
```

Panel E: Assertions on MCCO.PartFamily

```

(RootCtx.Type MCCO.PartFamily)
(RootCtx.argProp MCCO.PartFamily 1 RootCtx.Top)
(RootCtx.arity MCCO.PartFamily 1)
(RootCtx.name MCCO.PartFamily "PartFamily")
(RootCtx.rem MCCO.PartFamily "(PartFamily ?pf)")
(RootCtx.sup MCCO.DesignPartFamily MCCO.PartFamily)
(RootCtx.sup MCCO.PartFamily MCCO.Family)
(RootCtx.sup MCCO.ProductionPartFamily MCCO.PartFamily)

(integrityRule (RootCtx.fidEx "ProductLifecycleCoreConcepts.kif" 3)
(=> (MCCO.PartFamily ?pf)
(exists (?pt)
(and (MCCO.Part ?pt)
(MCCO.hasCriteria ?pf ?pt))))))

(integrityRule (RootCtx.fidEx "ProductLifecycleCoreConcepts.kif" 6)
(=> (MCCO.PartFamily ?pf)
(exists (?pt)
(and (MCCO.Part ?pt)
(MCCO.hasCriteria ?pf ?pt))))))

(=> (MCCO.PartFamily ?x) (MCCO.Family ?x))
(=> (MCCO.PartFamily @Args) (MCCO.Family @Args))
(=> (MCCO.DesignPartFamily ?x) (MCCO.PartFamily ?x))
(=> (MCCO.DesignPartFamily @Args) (MCCO.PartFamily @Args))
(=> (MCCO.ProductionPartFamily ?x) (MCCO.PartFamily ?x))
(=> (MCCO.ProductionPartFamily @Args) (MCCO.PartFamily @Args))
  
```

Panel F: Instances of MCCO.Family

- MCCO.ShaftFamily
- MCCO.BladeFamily
- MCCO.DiscFamily
- MCCO.BoltFamily
- MCCO.NutFamily

Figure 8.2: Implementation of the MCCO in IODE

figure 8.2 to demonstrate the successful deployment of those categories of concepts. The portion [A] of figure 8.2 shows the eight categories of concepts and their hierarchies. The prefixes attached to the concepts represent contexts. For example, the prefix 'MCCO.' represents that the context of the concepts is the MCCO. It will be shown later that the specialised ontologies will have different contexts. Next, the concept 'Family' is explored in detail to show the different aspects of the MCCO. The general information about each concept can be viewed through the 'Description' tab [B]. This table also shows the remarks, which describe the semantics of the selected concept in simple text. The relations of each concept can be browsed as well e.g. the relation defining the concept *Family* is *hasCriteria* which is shown along with its arguments under the tab 'Relations' [C]. All other relations involving the concept family can also be browsed by clicking on the '+' signs. Portion [D] shows that the

loaded concepts can also be viewed graphically. One of the most useful browsing capabilities offered by IODE is to check the heavyweight logic behind each concept. For instance, the set of axioms used to define the concept *PartFamily* are shown browsed under the 'Assertions' tab [E]. Another useful component of browsing is to view the facts that have been asserted. For example, the 'Instances' tab shows the facts asserted for the selected concept 'PartFamily' [F].

All the MCCO concepts have been successfully loaded into IODE as an XKS. Hierarchies of the eight categories of the MCCO concepts have been shown. This section also shows the description of concepts and their formalisations using the *Family* concepts. In a similar way, concepts from other categories can be illustrated.

8.4 Overview of the Experiments

A number of experiments have been identified to verify different research aspects presented in this thesis. An overview of those experiments is presented in Figure 8.3.

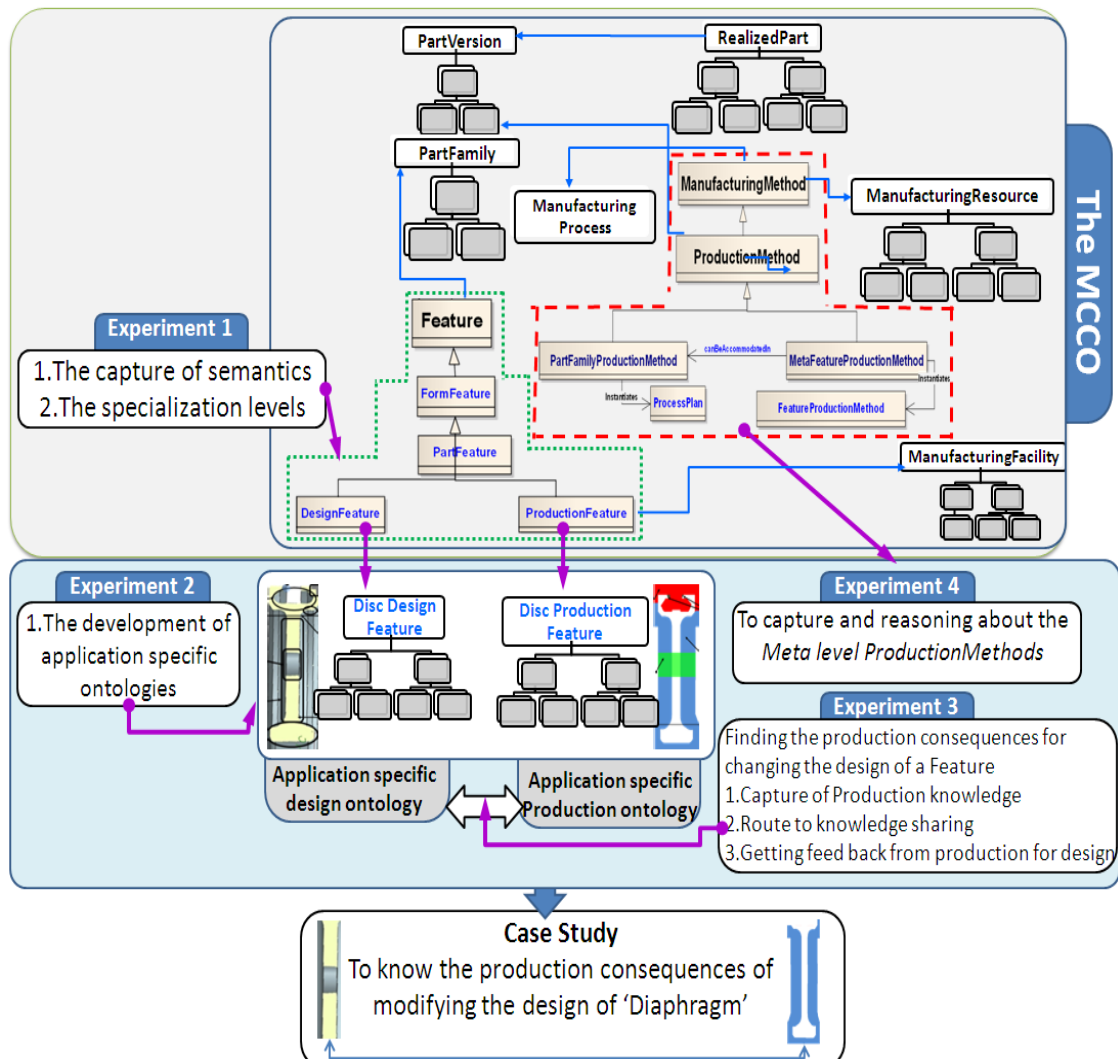


Figure 8.3: The arrangement of experiments

- Experiment 1: To verify the formal capture of semantics of concepts (against section 5.5 of chapter 5) and to verify the specialization levels for their ability to capture the varying depths of meaning of concepts (against section 6.4 of chapter 6).
- Experiment 2: The ability of the MCCO to support the development of semantically sound application specific ontologies for product design and production domains (against section 6.5 chapter 6).
- Experiment 3: To know the production consequences of changing the design which, involves the verification of (1) the capability of the MCCO to support the capture of production knowledge and (2) the provision of a route to knowledge sharing between product design and production domains (against section 6.6).
- Experiment 4: The verification of the adopted methodology and concepts to capture and reason about the *Meta level production knowledge* (against chapter 7).

In the end, a case study is presented to further strengthen the verification of the research concept.

8.5 Experiment1: Testing the Specialization Levels and the Capture of Semantics

8.5.1 Objective

This experiment aims to achieve the following two objectives.

1. To verify the capture of the semantics of the concepts.
2. The verify the use of specialization levels to capture the varying depths of meaning of the concepts.

8.5.2 Explanation

The capture of the semantics of concepts is verified as a part of the verification of the specialisation levels to capture the varying depths of meanings of concepts. The specialisation levels were explained using the *Feature* concepts (section 6.4). This experiment is conducted using the *Feature* concept and its various specializations. The testing of the semantics of either the *ProductionFeature* or the *DesignFeature* can verify the objectives of this experiment. However, the concept *ProductionFeature* is selected because of the focus on the production domain. The verification will come as results of the answers to the following questions.

1. Does the system allow the assertion of facts in violation of the definition of concepts?

2. Does the system report the violation of definitions from all the relevant specialization levels when the facts at a more specialized level like Design or Production are asserted?
3. Does the system report the requirements that need to be specified for a fact to meet the formal definitions of concepts?
4. Does the system allow the assertions when the asserted facts satisfy the definitions of the specific concepts and other concepts involved in its specialization?

8.5.3 Procedure

A fact for *ProductionFeature* is first asserted without a production method as its attribute of interest and then with a production method. Figure 8.4 shows the assertion of a *ProductionFeature*'s fact without a *ProductionMethod* as its required attribute of interest. The upper box shows the fact assertion and the lower box 'Assertion Log' shows the results of the assertion.

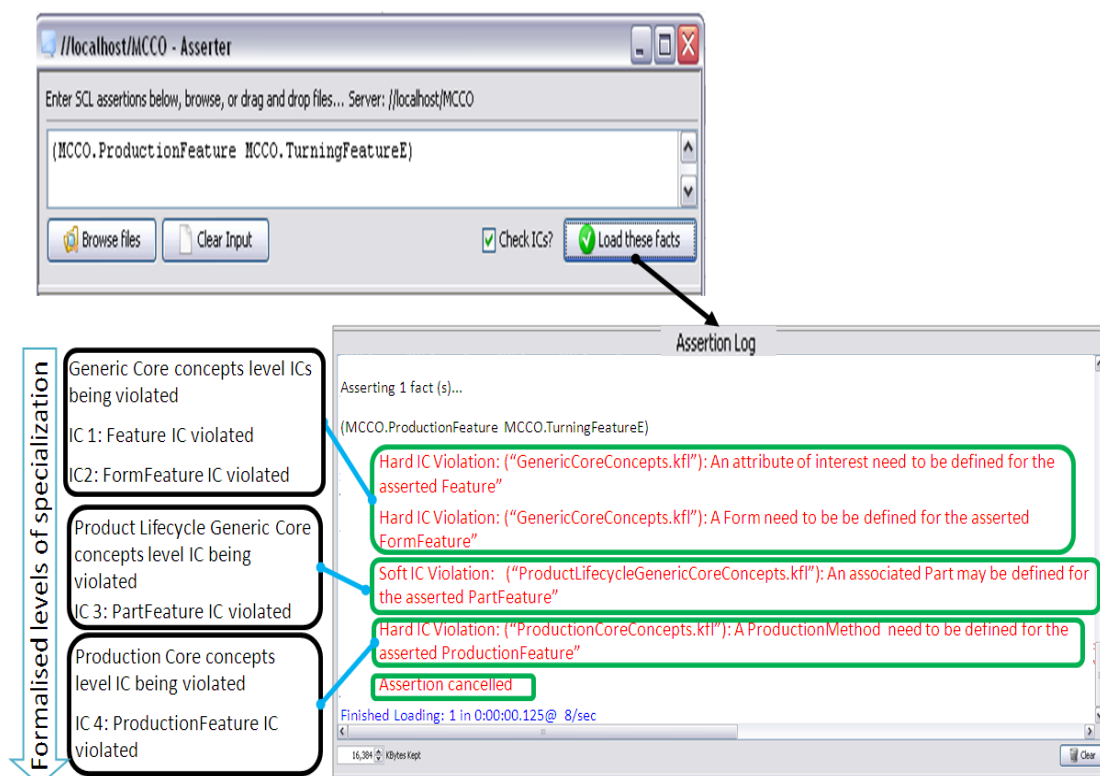


Figure 8.4: Asserting a ProductionFeature without its ProductionMethod

Figure 8.5 shows the assertion of the same fact (*TurningFeatureE*) with its *ProductionMethod* (*Turning*) as the required attribute of interest. Moreover, the *Form* (*FormE*) and the associated *Part* (*Disc*) were also asserted with '*TurningFeatureE*'.

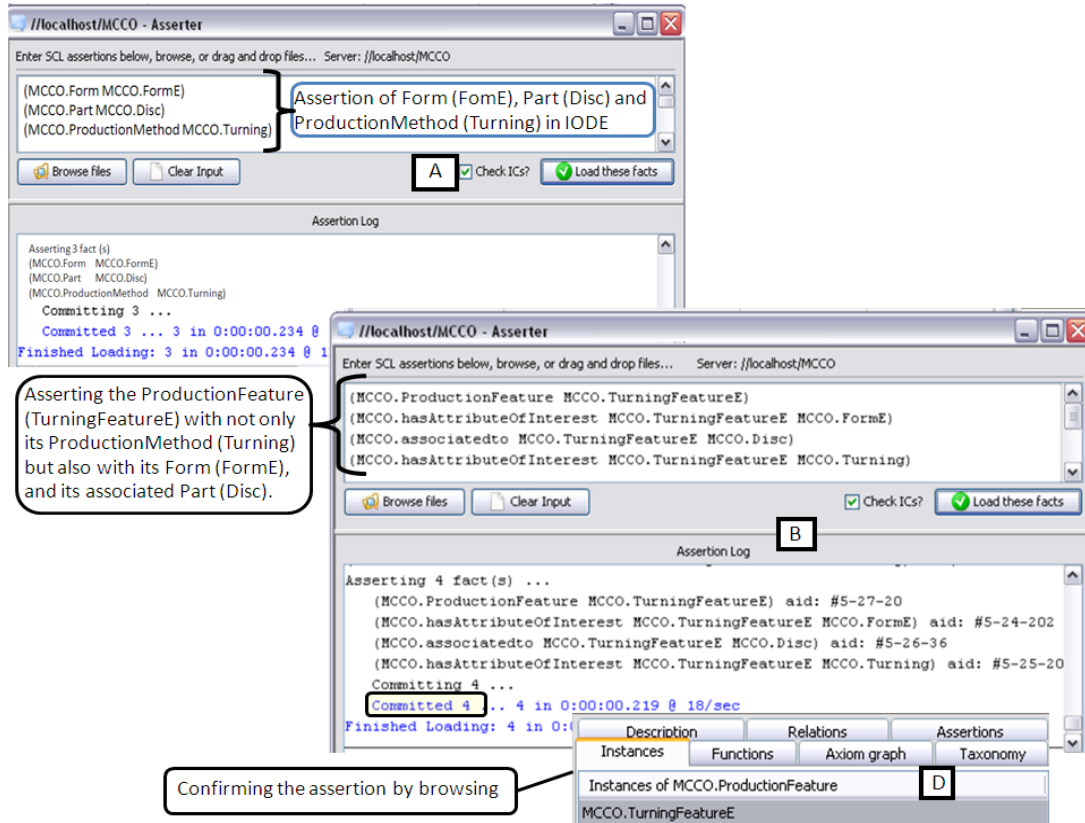


Figure 8.5: Asserting a ProductionFeature without its ProductionMethod as well as Form and associated Part

Similar to the assertion of facts for *ProductionFeature*, experiments were conducted for other *Feature* concepts at the three levels of specialisation. Table 8.1. lists these

Table 8.1: Results of assertions for various Feature concepts

Sr	Assertion	Result of Assertion			
		✗	ICs violated in cancellation '✗' of assertion		
		✓	Specialization Level	Type	IC message
1	Feature without Attribute of Interest	✗	IC1.Generic Core Level	Hard	IC1: An attribute of interest need to be defined for the asserted Feature
	Feature with Attribute of Interest	✓			
2	FormFeature without Form	✗	IC1.Generic Core Level IC2.Generic Core Level	Hard Hard	IC1: An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature
	FormFeature with Form	✓			
3	PartFeature without Part	✗	IC1.Generic Core Level IC2.Generic Core Level IC3.ProductLifecycle Core Level	Hard Hard Soft	IC1 : An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature IC3:An associated Part may be defined for the asserted PartFeature
	PartFeature with Part and Form	✓			
4	DesignFeature Without Function	✗	IC1.Generic Core Level IC2.Generic Core Level IC3.ProductLifecycle Core Level IC4a: Design Core Level	Hard Hard Soft Hard	IC1: An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature IC3: An associated Part may be defined for the asserted PartFeature IC4a: A Function need to be defined for the asserted DesignFeature
	DesignFeature With Function, Part and Form	✓			
5	ProductionFeature With out ProductionMethod	✗	IC1.Generic Core Level IC2.Generic Core Level IC3.ProductLifecycle Core Level IC4b: Production Core Level	Hard Hard Soft Hard	IC1: An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature IC3: An associated Part may be defined for the asserted PartFeature IC4b: A ProductionMethod need to be defined for the asserted ProductionFeature
	ProductionFeature With ProductionMethod , Part and Form	✓			

assertions and their results. Table 8.1 states that ICs 1, 2 and 4 are hard whereas, IC

3 is soft. The ICs 1 and 2 refer to the generic core concepts level, IC3 refers to the product lifecycle level and IC4a and IC4b refer to design and production core levels respectively. A discussion on these results is given next.

8.5.4 Discussion on Results

On asserting the *ProductionFeature* and *DesignFeature* facts without their respective *ProductionMethod* and *Function*, the assertions were cancelled because a number of hard ICs were violated as shown in figure 8.4 and rows 4 and 5 in table 8.1. As shown in table 8.1, IC4a for the *DesignFeature* caused the cancellation of asserted *DesignFeature* fact and reported the requirements, in 'IC message', that need to be satisfied for the successful assertion of facts. Similarly, IC4b for the *ProductionFeature* caused the cancellation of asserted facts and reported the requirements, in 'IC message'. The IC messages reported that a *ProductionMethod* need to be defined for the asserted *ProductionFeature* and a *Function* need to be defined for asserted *DesignFeature*. It shows that the system does not allow the assertions which, violate the formal definitions and also specifies the requirements that need to be satisfied in accordance with the formal definitions of concepts.

However, IC4a and IC4b were not the only ICs being violated. The ICs from generic core concepts level (IC1 and IC2) for the concepts *Feature* and *FormFeature* as well as the IC from product lifecycle core level (IC3) for the concept *PartFeature* were also violated. The requirements to satisfy the definitions of concepts from these levels were reported in the messages of ICs1, 2 and 3. Thus, the system directs that the assertion for *DesignFeature* and *ProductionFeature* facts should be made while satisfying the definitions of *Feature*, *FormFeature* and *PartFeature* concepts as well.

When the same *ProductionFeature* fact 'TurningFeatureE' was asserted with its *ProductionMethod*, its *Form* and its associated *Part*, the assertion was a success and no ICs were violated (figure 8.5). The assertions of *Form* and *Part* with 'TurningFeatureE' satisfied the definitional requirement of *Feature*, *FormFeature* and *PartFeature*. The confirmation of assertion is done by the message stating 'committed' and also through browsing the instances of *ProductionFeature* as shown in figure 8.5[D]. Similarly, the assertions have been successful when the requirements for all the concerned specialization levels have been satisfied (table 8.1).

Another important point that can be noted from figure 8.5, is regarding the reuse of the already asserted facts. As depicted in figure 8.5 [A], initially only the facts FormE, Disc and Turning were asserted. Then these were reused (figure 8.5 [B]) when

asserting the 'TurningFeatureE'. This highlights the reusability of asserted knowledge. These results and the discussion on them help to draw the following conclusions.

8.5.5 Conclusions

1. The system has the capability to understand the semantics of concepts and directs the user to follow the formal definitions.
2. The specialization approach has enabled the knowledge system to formally capture and understand the gradually varying depths of meaning of concepts.
3. The semantics are consistent throughout the specialisation levels and the depths of meaning is inherited by the Design and Production specific core concepts from the generic level and product lifecycle generic level concepts.
4. The specialised core concepts cannot violate the definitions of the concepts from which they are specialised.
5. Asserted facts can be reused to formulate new facts and production knowledge. This is established from the reuse of facts shown in figure 8.2([A] and [B]).

8.6 Experiment 2: Developing Application Specific Ontologies using the MCCO

8.6.1 Objectives

- To verify the ability of the MCCO to support the development of application specific product design and production ontologies.
- To verify the semantic integrity of the application specific ontologies developed based on the MCCO.

8.6.2 Explanation

This experiment verifies the claims made in section 6.5. This experiment will mainly extend the *Feature* concepts from the MCCO to develop the application specific product design and production ontologies. However, other concepts from the MCCO will also be used to support this development.

8.6.3 Procedure

The disc design and production views of the studied disc (figure 3.6) are taken as the application specific product design and production ontologies. These are named as 'AeroEngineDiscProductionOntology' and 'AeroEngineDiscDesignOntology'. The procedure consists of identifying the concepts for these ontologies and then formalising them in lightweight and heavyweight logic. These ontologies are then implemented in the IODE to test their semantic integrity.

8.6.4 Formalisation and Implementation of Application Specific Ontologies

The identification of concepts for the application specific ontologies is based on the industrial investigation of the studied disc presented in chapter 3. The lightweight formalizations of these concepts for the AeroEngineDiscDesignOntology and AeroEngineDiscProductionOntology are presented in figure 8.6. These lightweight ontologies capture all the disc design and production features which were identified in chapter 3. The figure also shows the use of these ontologies to capture the aero engine disc design and production views.

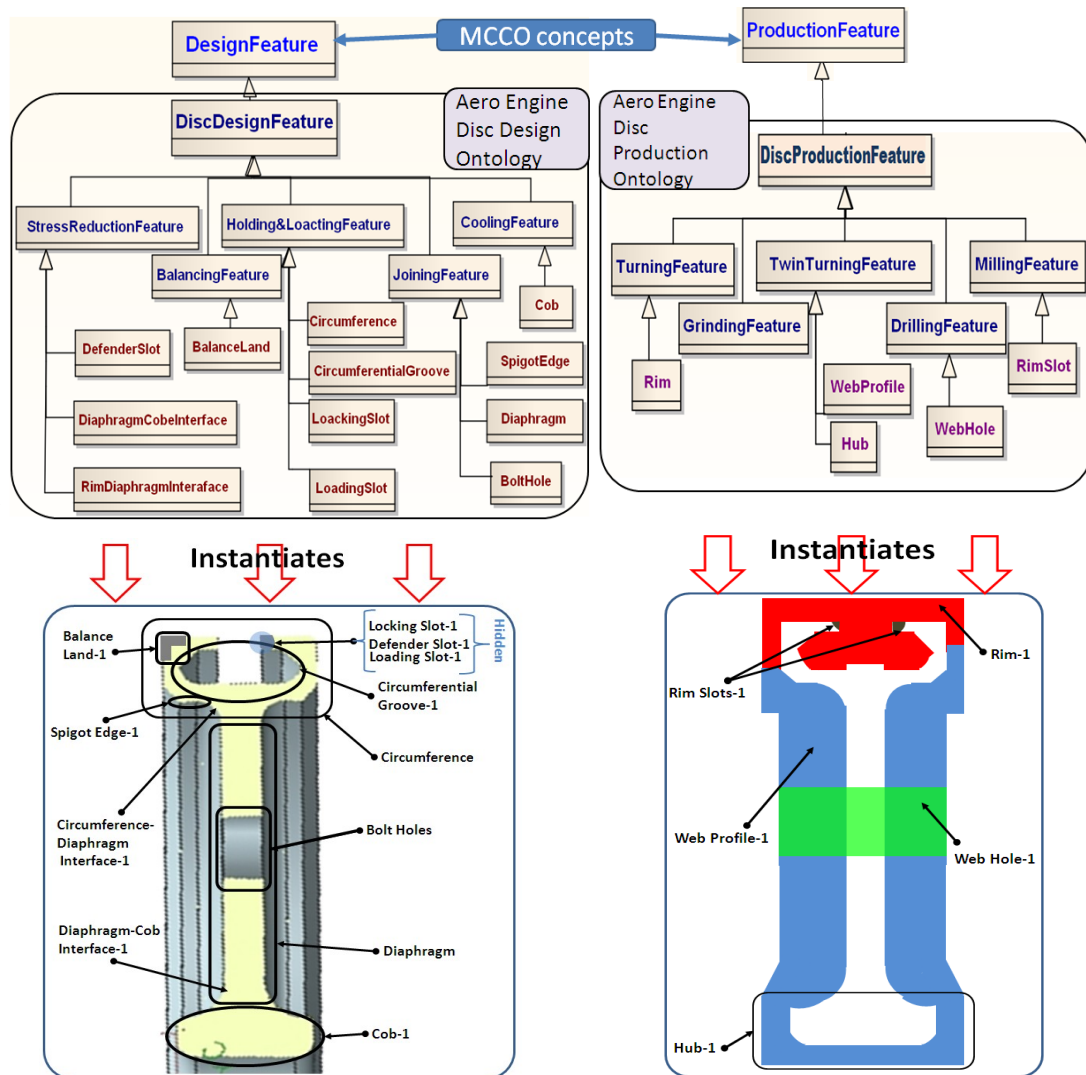


Figure 8.6: Lightweight view of application specific ontologies developed from the MCCO and the instantiated disc design and production views of the application specific ontologies.

The important point is that these ontologies have been developed using the core concepts from the MCCO which provide the semantic integrity to these ontologies. Figure 8.7 shows the application specific product design and production ontologies being implemented in IODE as XKSs. This figure also shows the description and

heavyweight formalisation of application specific ontologies. Figure 8.7 also shows the hierarchies of concepts for AeroEngineDiscProductionOntology and AeroEngineDiscDesignOntology. The context ‘.MCCO’ represents the MCCO, the

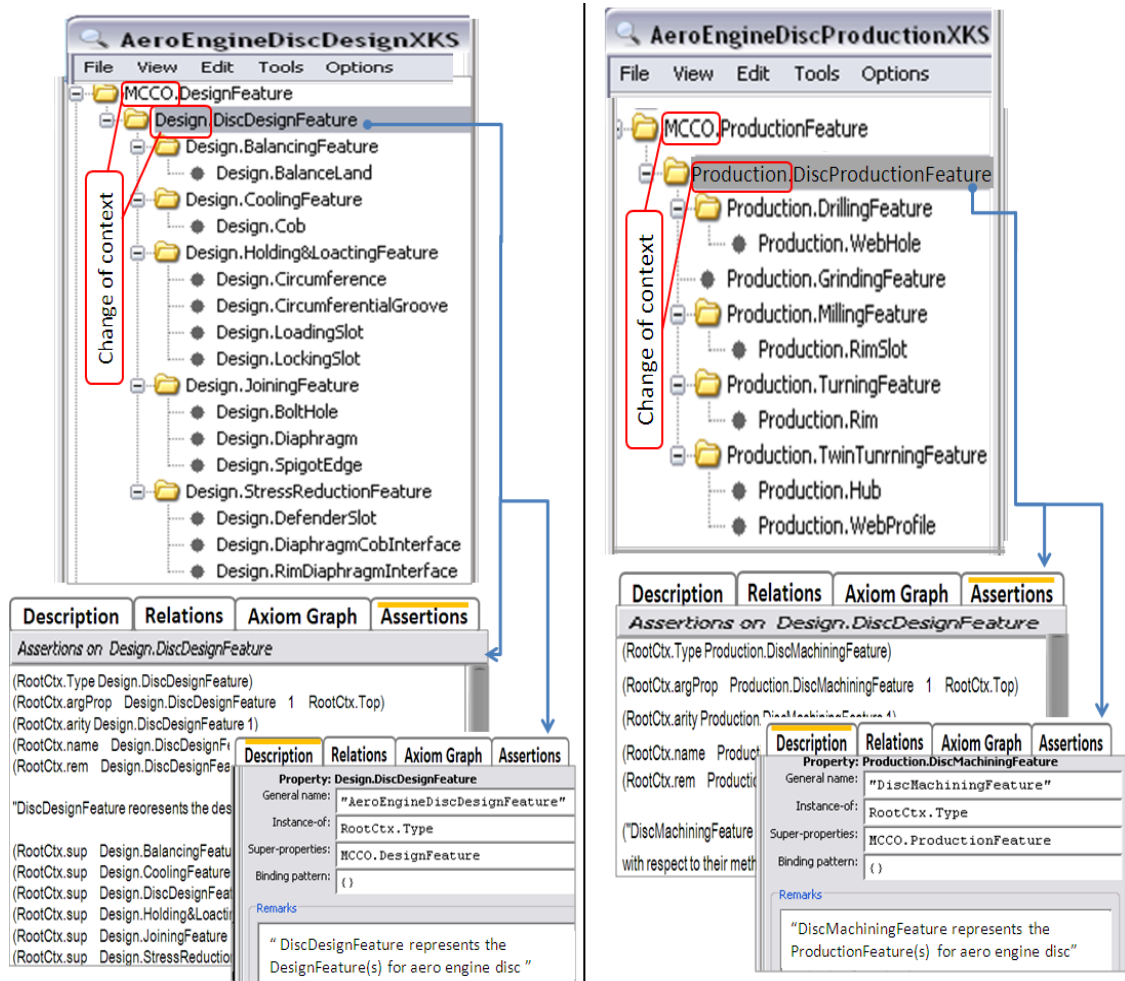








Figure 8.7: Implementation of the application specific ontologies, developed using the MCCO, in IODE context ‘.Production’ represents the ‘AeroEngineDiscProductionOntology’ and the context ‘.Design’ ‘AeroEngineDiscDesignOntology’.

8.6.4.1 Verification of the Semantics in the Application Specific Ontologies

The successful implementation is a step towards proving the development of application specific ontologies using the MCCO. However, it is not enough to verify the integrity of the semantics inherited in the application specific ontologies from the MCCO.

In order to test this, the facts for concepts like ‘DiscDesignFeature’ and ‘DiscProductionFeature’ from the application specific ontologies are asserted in the XKS. The procedure is the same as that of experiment 1. The assertions and their results are listed in table 8.2.

Table 8.2: The results of assertion for application specific ontologies

Sr	Assertion	Result of Assertion			
		 	ICs violated in 'X'		
			Specialization Level	Type	IC message
1	Disc DesignFeature	Without any attribute	 IC1.Generic Core Level IC2.Generic Core Level IC3.ProductLifecycle Core Level IC4a: Design Core Level	Hard Hard Soft Hard	IC1: An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature IC3: An associated Part may be defined for the asserted PartFeature IC4a: A Function need to be defined for the asserted DesignFeature
		With Function,Part and Form			
2	DiscProductionFeature	With out ProductionMethod	 IC1.Generic Core Level IC2.Generic Core Level IC3.ProductLifecycle Core Level IC4b: Design Core Level	Hard Hard Soft Hard	IC1: An attribute of interest need to be defined for the asserted Feature IC2: A Form need to be defined for the asserted FormFeature IC3: An associated Part may be defined for the asserted PartFeature IC4b: A ProductionMethod need to be defined for the asserted ProductionFeature
		With ProductionMethod, Part and Form			

8.6.5 Discussion on Results

Regarding figure 8.7, it is important to mention that the context of the application specific ontologies changes from 'MCCO.' to 'Design.' and 'Production.' when their own concepts start to appear in the hierarchy as shown in figure 8.7. This shows that the application specific ontologies are developed using the MCCO. The figure not only shows the implementation of application specific ontologies but also presents the browsing of their heavyweight formalisation. The verification of semantics based on that formalisation comes from the results presented in table 8.2.

As shown in table 8.2, the assertions for both application specific feature concepts i.e. 'DiscDesignFeature' and 'DiscProductionFeature' were made in the XKS. These assertions were cancelled when no attribute was defined for the asserted facts. This happened due to the violation of number of hard and soft ICs. All the ICs that have been violated, belong to the MCCO. Similar to the experiment 1, when the same facts were asserted after taking care of the ICs, i.e. facts were asserted with their forms, parts, function and production methods, the assertions were committed to the XKS (table 8.2).

8.6.6 Conclusion

The following conclusions are drawn from the results and discussion of experiment 2:

- The MCCO supports the development of application specific production design and production ontologies.
- The application specific product design and production ontologies inherit the semantics of the MCCO and thus possess semantic integrity.
- The application specific product design and production ontologies can be developed within their own contexts while still being committed to the MCCO.

The next experiment is performed to show the production consequences of changing the design of a *DesignFeature* by utilising the semantic base provided by the MCCO.

8.7 Experiment 3: To Show the Production Consequences of Changing the Design of a Feature

This experiment is aimed to verify the following research objectives

1. The MCCO supports the capture of production knowledge
2. The MCCO provides the route to share knowledge between product design and production domains

8.7.1 Procedure

In order to verify the above objectives an example case study based on the 'CircumferentialGroove' from the AeroEngineDiscDesignOntology and its corresponding *DiscProductionFeature* 'Rim' from AeroEngineDiscProductionOntology is taken. This experiment consists of the following steps;

1. Capturing the production knowledge:
2. Establishing the route to knowledge sharing between disc design and production
3. Getting the feedback from production into design

8.7.1.1 The Capture of Production Knowledge

As explained in figure 3.5 of chapter 3, *DiscProductionFeature* 'Rim' can be machined with the available tooling provided the value of its neck width and groove angle are satisfying the rules 1.1 to 1.3 reported in figure 8.8. The formalisation of this

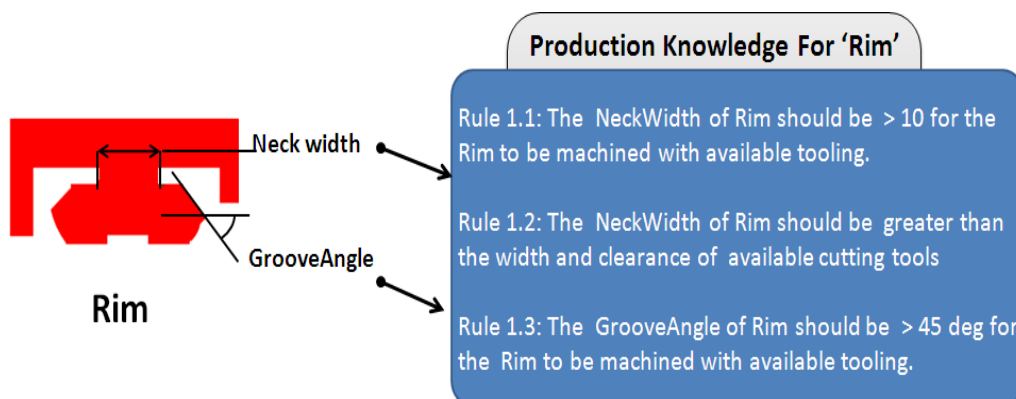


Figure 8.8: The knowledge to be captured for the Rim

knowledge in axioms requires the following relations and functions listed in table 8.3.

Table 8.3: The relations and function for capturing Rim's production knowledge

Relation	Corresponding variables	Functions	Corresponding variables
<i>hasParameter</i>	To define parameters of a <i>Form</i> To define parameters of a <i>CuttingTool</i>	mm	For Length
<i>hasValue</i>	To define values of Parameters	deg	For angle
<i>gteNum</i> (from ULO)	To relate a bigger value with a smaller one		
<i>hasToolDiaAndClearanceValue</i>	To infer the combined values of <i>CuttingsTool</i> 's diameter and clearance		
<i>canMachine</i>	To relate <i>CuttingTool</i> to the parts that <i>canmachine</i>		

The relations and functions presented in the above table are then used in axioms to formally capture the knowledge. The first of these axioms is stated below;

;Rim's Production Knowledge Axiom 1

```
(=> (and (Production.Rim ?rim)
          (MCCO.Form ?groove)
          (MCCO.hasAttributeOfInterest ?rim ?groove)
          (MCCO.hasParameter ?groove ?nw)
          (MCCO.hasValue ?nw (MCCO.mm ?real1)))
      (gteNum ?real1 10))
```

:IC hard "The NeckWidth of Rim should be greater than 10mm for the Rim to be machined with standard tooling"

The above axiom will disallow the assertion of any *Rim* fact whose *NeckWidth* is less than 10mm and will also notify the user about this requirement. However, there can be a condition where the *NeckWidth* is greater than 10mm but it is still smaller than the diameter and clearance of the *CuttingTool* used. Therefore, the following rule evaluates the *NeckWidth* of Rim in relation to its machining.

;Rim's Production Knowledge Axiom 2

```
(=> (and (Production.Rim ?rim)
          (MCCO.hasAttributeOfInterest ?rim ?groove)
          (MCCO.hasValue ?nw (MCCO.mm ?real1))
          (MCCO.hasToolDiaAndClearanceValue ?ct (MCCO.mm ?real2))
          (MCCO.canMachine ?ct ?nw))
      (gteNum ?real1 ?real2))
```

:IC hard "The NeckWidth of Rim should be greater than diameter and clearance values of the CuttingTool for the Rim to be machined with standard tooling"

This axiom uses the inferred value of the summation of diameter and clearance values of the *CuttingTool*(s) from the XKS. The above rule is very powerful in the sense that it can compare the values of the *NeckWidth* and *GrooveAngle* of the asserted *Rim* fact against the diameters and clearances of the available *CuttingTool*(s) and can feed back the manufacturability knowledge.

An axiom 'Rim's Production Knowledge Axiom 3' similar to the 'Rim's Production Knowledge Axiom 1' is defined to capture the knowledge which can be stated as; "The GrooveAngle of Rim should be greater than 45deg for the *Rim* to be machined with standard tooling".

8.7.1.2 Establishing the Route to Knowledge Sharing

The knowledge captured in axioms presented in section 8.7.1.1 is embedded within the XKS and can be shared with the relevant *DesignFeature(s)*. 'CircumferentialGroove' is one such *Feature* from the *AeroEngineDiscDesignOntology*. Consider the case where a designer changes the *NeckWidth* value of 'CircumferentialGroove' to 6mm and the *GrooveAngle* to 30deg. An instance of 'CircumferentialGroove' is asserted by the name 'CircumGroove1' with *NeckWidth* =6mm and *GrooveAngle* = 30deg in the XKS. The assertion can be successfully made in the design KB. However, if after these changes, the designer intends to know the production consequences, this can be possible if the *ProductionFeature* relevant to the 'CircumGroove-1' can be identified. Because the AeroEngineDisc's design and production ontologies are based on the MCCO, the MCCO should provide the route to identify the relevant *DiscProductionFeature* and thus a route to share knowledge.

It has been explained in section 3.3.2 of chapter 3 and section 6.6 of chapter 6 that different design and production features can be related by their overlapping forms through the *FormFeature* concept. It is important to mention here that common forms do not mean that design and production features have same form but, they can have overlapping portions of their forms which can help to relate the two different features. For example, when a query was made in the XKS to identify any *DiscProductionFeature* that has same or similar *Form* as that of the 'CircumGroove1'. The results showed (figure 8.9 portion B) that CircumGroove1 is related to Rim1 through their common form but, Rim1 also links to 'BalanceLand1' through their

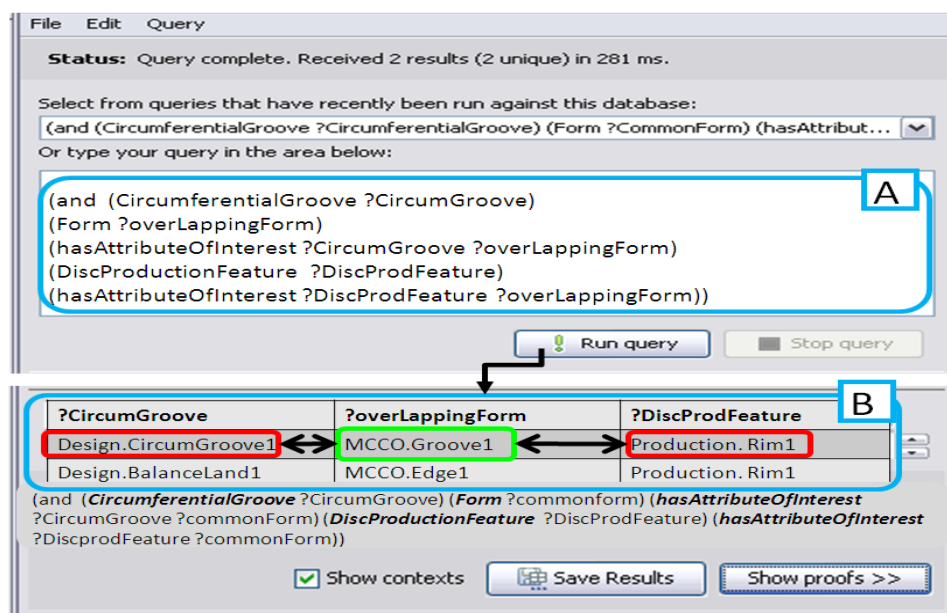


Figure 8.9: The query to link Disc Design and Production Features through the MCCO

common form Edge1. This shows that form of Rim1 is overlapping the forms of CircumGroove1 and BalanceLand1 and the forms identified as common forms are the overlapping portions of the forms of design and production features.

On running the query, the systems identifies the results [B] which state that the form of *DiscDesignFeature* 'CircumGroove1' is encompassed within the form of *DiscProductionFeature* 'Rim1'. These results are based on the semantics inherited from the concepts *FormFeature* of the MCCO. The results of this query has established a route to link the 'CircumGroove1' to its corresponding disc production feature 'Rim1' and thus provided a route to share knowledge.

8.7.1.3 Getting Feedback from Production into Design

After the establishment of knowledge sharing route, the next step is to get the production feedback on the designed values of 'CircumGroove1'. This is done by asserting the designed values of *NeckWidth* and *GrooveAngle* of CircumGroove1' in the AeroEngineProduction KB for the 'Rim1'. The results of the assertion of Rim1 with the designed values of *NeckWidth* and *GrooveAngle* constitute the feedback for 'CircumGoove1'. Therefore, an assertion is made for an instance of Rim 'Rim1' with the NeckWidth =6mm and GrooveAngle =30deg as shown in figure 8.10.

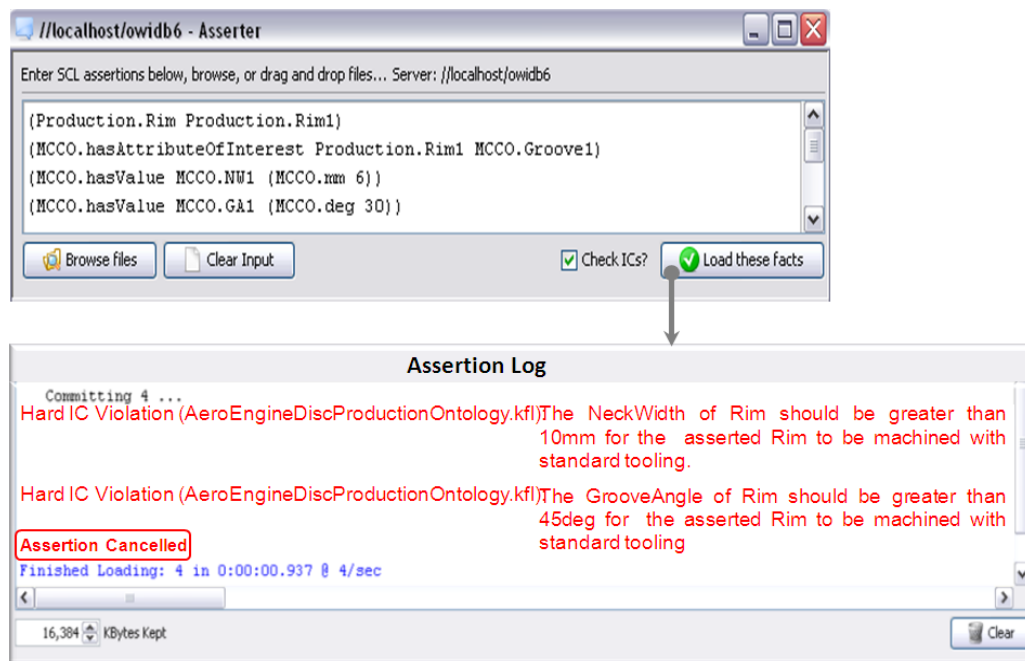


Figure 8.10: Assertion of a Rim fact 'Rim1' with NeckWidth=6mm and GrooveAngle=30deg
 As shown in the figure the assertion was cancelled as a result of the violation of ICs from the AeroEngineDiscProductionOntology. The messages of the violated ICs constitute the production knowledge feedback for the design.

The first IC message states that the NeckWidth should be greater than 10mm for the asserted Rim and the second IC message states that the GrooveAngle should be greater than 45deg for the asserted Rim to be machined with standard tooling. These ICs fired as a result of the production knowledge being formally captured in the XKS. The feedback from the production knowledge to the designer is composed of these IC messages.

Based on this feedback the designer can modify the value of the NeckWidth and the GrooveAngle and re-evaluate the design. Assume that the designer changed the values of NeckWidth and GrooveAngle to 10 mm and 45 deg respectively. The link of the CircumferentialGroove's instance 'CircumGroove1' has already been established with the corresponding instance 'Rim1' of feature *Rim* from AeroEngineDiscProductionOntology. Therefore, a 'Rim1' is re asserted with the new value of the NeckWidth and GrooveAngle. Because this assertion does not violate the production knowledge ICs, the assertion has been successful as shown in the figure 8.11. As shown in the table the assertion was a success this time and none of the production ICs were violated. This means that the present design can be produced.

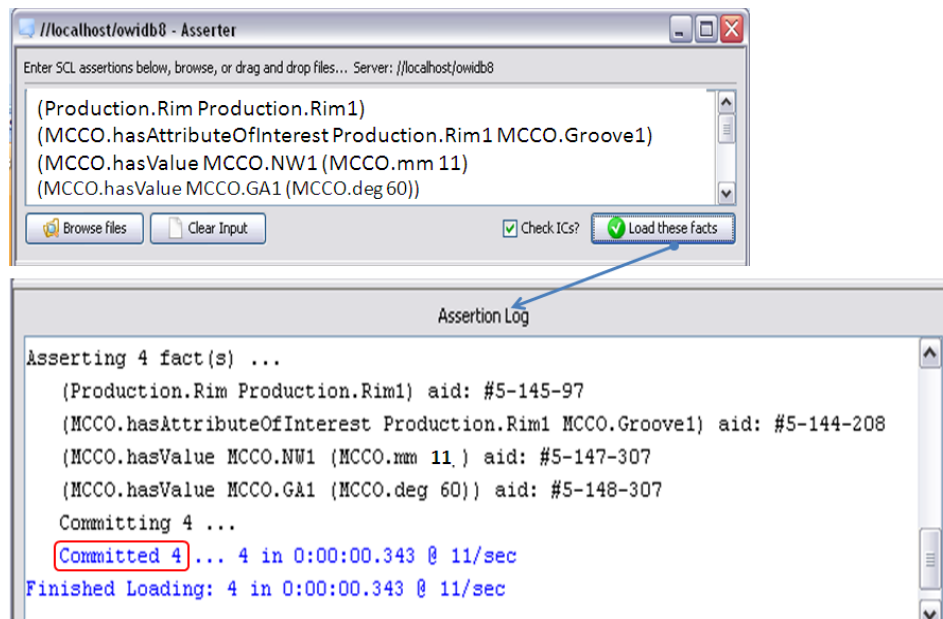


Figure 8.11: Assertion of Rim fact 'Rim1' with NeckWidth =11mm and GrooveAngle =60 deg

8.7.2 Discussion on Results

Steps 1, 2 and 3 in of this experiment, reported in sections 8.7.1.1-8.7.1.3, have successfully shown the ability of the MCCO to support the capture of production knowledge in an application specific domain, the provision of route to knowledge sharing and the feedback from the production knowledge to the design. The step one of capturing the knowledge has to be done manually as it involves the formal coding

of production knowledge. However, steps 2 and 3 can be automated where, based on the formalisation, the system can itself pick the design feature, identify its relevant production feature, asserts the values of design feature in production and feedback the results to the designer. This has been practically implemented, tested and demonstrated in the IMKS project.

The establishing of the route to knowledge sharing through the forms of *DesignFeature(s)* and *ProductionFeature(s)* are not the same but they can have overlapping portions of forms that can help establish the link between the *DesignFeature(s)* and *ProductionFeature(s)*. It is evident from the results and discussion that;

- The core concepts support the capture of production knowledge for application specific production domains.
- The MCCO provides a route to knowledge sharing by providing a semantic base to link the application specific product design and production ontologies.
- The production knowledge can be fed back to the designer in the form of IC messages of the violated ICs.

8.8 Experiment 4: Testing the Capture of and Reasoning about the Meta Level ProductionMethods

8.8.1 Objectives

This experiment is built as a proof of the structure to capture and reason about production knowledge that was reported in chapter 7 of this thesis. Objectives of this experiment are:

1. To verify the capture of Meta level feature and part family production methods
2. To verify the reasoning capability over the Meta level production methods

8.8.2 Overview and procedure

It is understood from chapter 7 that the Meta level knowledge is composed of clabjects. Clabjects (being concepts) contribute to the structure of the ontology. Therefore, the *Meta level knowledge* contributes to the structure of the ontology which, can make the already asserted facts inconsistent with respect to the modified structure. Therefore, the *Meta level knowledge* cannot be asserted like other facts and has to be asserted as part of the ontology structure at the time of loading the KFL files. For this reason, different Meta level *ProductionMethods* for *Feature(s)*, *Part(s)* and *PartFamily(s)*, have been captured through KFL files while deploying the XKS.

Meta level knowledge cannot be asserted like other facts but it can be queried like other facts. As a result, the experimental verification of the captured Meta level production methods and the reasoning about them is done through queries. The following queries will be made;

1. Query to acquire the Meta level PartFamily production method for StandAloneDisc Family (figure 8.12 [1])
2. Query to acquire the Meta level *FeatureProductionMethod* for Rim (figure 8.12 [2]).
3. Query to verify the manufacturability of a feature within a part family at Meta level (figure 8.12 [3]).

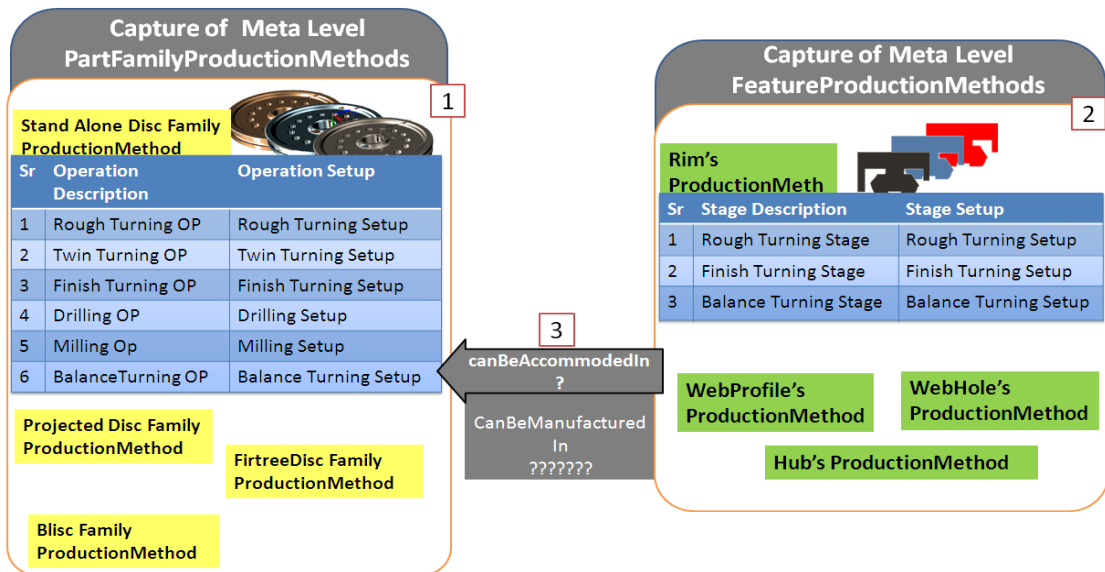


Figure 8.12: Overview of the Meta level production methods being captured

8.8.2.1 Acquisition of PartFamilyProductionMethod(s) at the Meta Level

The relation that captures the sequencing of *MetaOperations*, *MetaStages*, *MetaSteps* and their clabjects is *minPrecedes*. The queries will therefore, be made using this relation. The query and its results to acquire the Meta level ProductionMethod for the StandAloneDiscFamily is given shown in figure 8.13.

In figure 8.13, portion [1] shows the query where the first line of query acquires the sequence of events in the StandAloneDiscProductionMethod. The second and third lines specify that the sequencing of events being queried consists of clabjects that are instantiated from *MetaOperation*. The fourth line stipulates that the sequence belongs to clabject StandAloneDiscProductionMethod that is instantiated from *PartFamilyProductionMethod*. The portion [2] in figure 8.13 shows the acquired clabject of *PartFamilyProductionMethod* i.e. StandAloneDiscFamilyProductionMethod where the *MetaOperation's* clabjects involved are listed with respect to their sequence. The *Setup's* clabjects of all the operation's clabjects involved in the above production method have also been queried [3]. These queries and their results show

the ability of the MCCO to support the capture of the Meta level part family production methods.

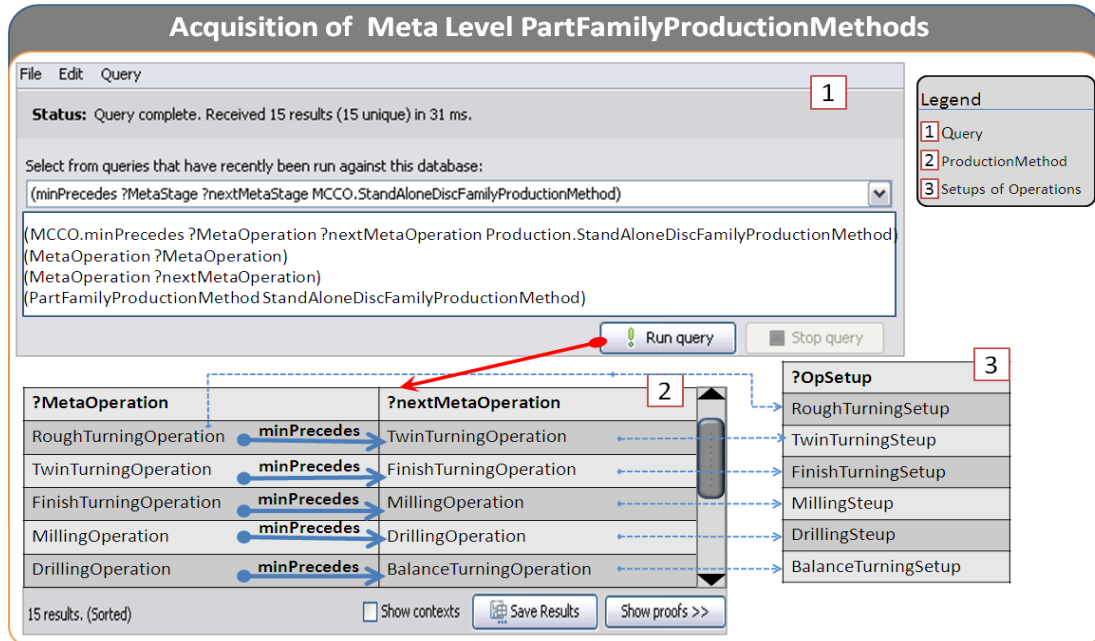


Figure 8.13: Acquisition of StandAloneDiscFamilyProductionmethod from the XKS

8.8.2.2 Acquisition of the FeatureProductionMethod at the Meta Level

Similar to the acquisition of the *PartFamilyProductionMethod*'s clabject in the last section, the clabject of *MetaFeatureProductionMethod* i.e. *RimProductionMethod* has also been queried as shown in figure 8.14. This shows the ability of the MCCO to support the capture of the Meta level feature production methods.

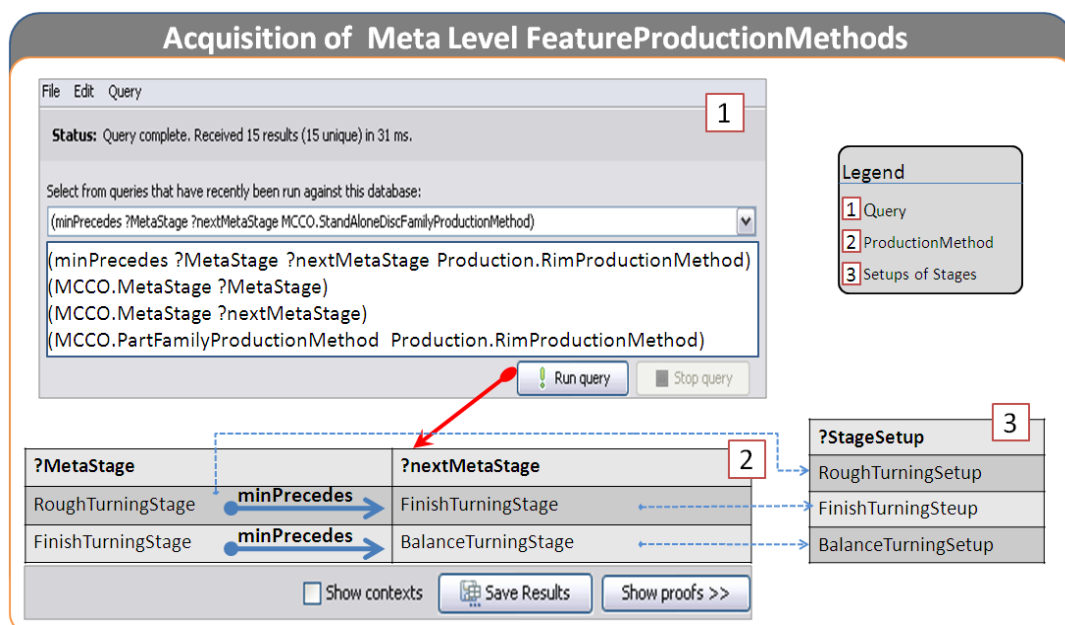


Figure 8.14: Acquisition of RimProductionmethod from the XKS

8.8.2.3 Reasoning about the Manufacturability of MetaFeatureProductionMethod in PartFamilyProductionMethod

In order to reason about the manufacturability of *MetaFeatureProductionMethod*(s) in *PartFamilyProductionMethod*(s), a binary relation *canBeAccommodatedIn* was defined in chapter 7. Different concepts and relations and their axiomatization that enables reasoning over the manufacturability of a clabjects of *MetaFeatureProductionMethod* within the clabjects of *PartFamilyProductionMethod* has already been detailed in section 7.3.3.

The Meta level production method queried in figure 8.13 and 8.14 i.e. *StandAloneDiscFamilyProductionMethod* and *RimProductionMethod* are not the only Meta level production methods that have been captured in the XKS. Several other Meta level production methods have also been captured. In the presence of several Meta level production method, can the system reason over the manufacturability of a certain clabject of *MetaFeatureProductionMethod* in the different clabject of *PartFamilyProductionMethod*. This is investigated by making the required query in the system. Figure 8.15 depicts the acquisition of *PartFamilyProductionMethod*'s clabjects in which the *RimProductionMethod* can be accommodated. As shown in figure 8.15, the *RimProductionMethod* can be accommodated in two clabjects of *PartFamilyProductionMethod* i.e. *StandAloneDiscFamilyProductionMethod* and *ProjectedDiscFamilyProductionMethod*.

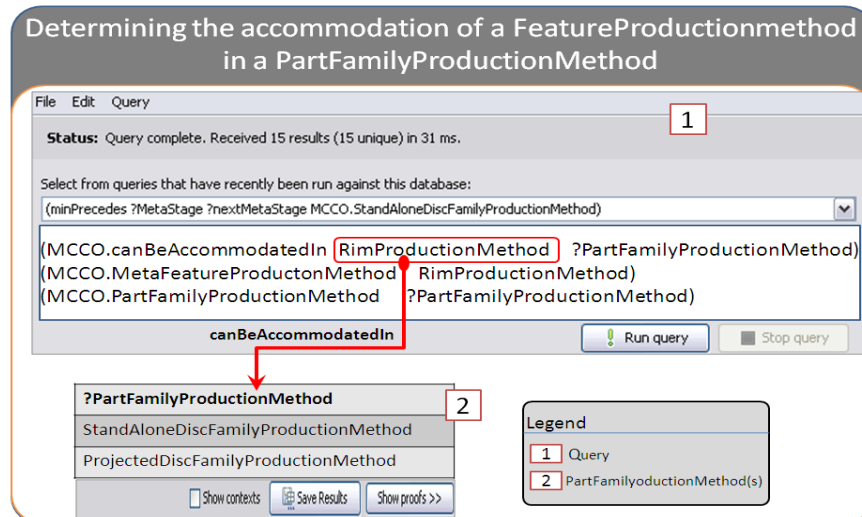


Figure 8.15: Query for determining the accommodation of a FeatureProductionmethod in a PartFamilyProductionMethod

The complex logic working at the backend of this query has already been explained in section 7.3.3. As mentioned in section 7.3.3, before the system can answer the manufacturability of a *MetaFeatureProductionMethod*'s clabject in a *PartFamilyProductionmethod*'s clabject, the manufacturability of the stages of

MetaFeatureProductionMethod's clabject in the operations of the *PartfamilyProductionProductionMethod*'s clabject has to be known.

This is done automatically within the system based on the complex logic explained in section 7.3. However, if it is desired to find out the manufacturability of the stages in a particular operation, this can be done as illustrated in figure 8.16 which shows the query to find out the manufacturability of stages of a *FeatureProductionMethod* in the

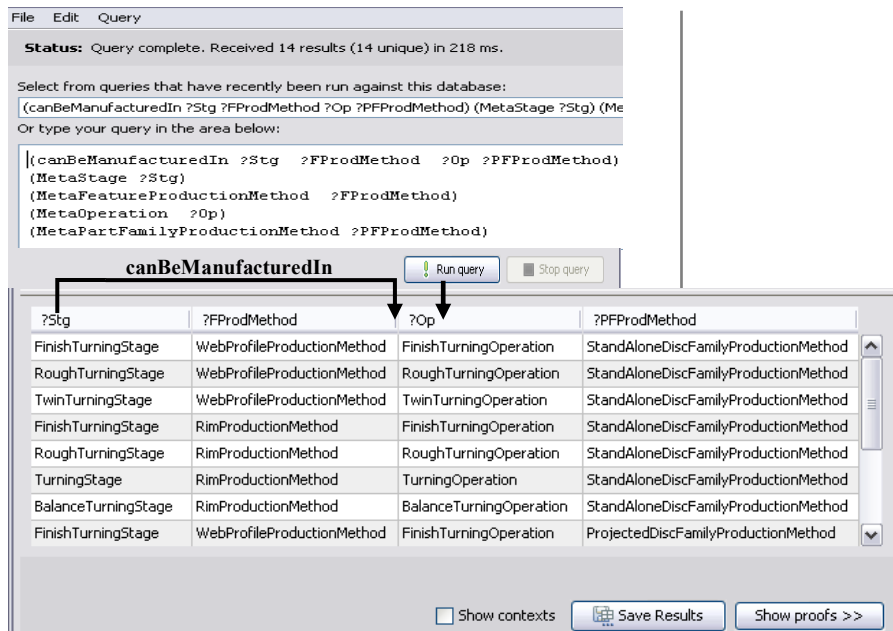


Figure 8.16: To find out the Manufacturability of Stages in Operations

operations of *PartFamilyProductionMethods* at the Meta level. The first line of query constitutes the main reasoning. Using only the first line, knowledge form both *individual* and *Meta levels* will be acquired. The next four lines specify that the query is to be made for the *Meta level knowledge* only. Results of the query list the *Stages* (of *FeatureProductionMethods*) and the *Operations* (of to *PartFamilyProductionMethods*) in which those *Stages* can be manufactured. This verifies the ability of the MCCO to support reasoning over the manufacturability of stages in operations at the meta level.

8.8.3 Discussion and Conclusions

This experiment shows the knowledge capture and reasoning has been made possible at the *Meta level knowledge*. The acquired sequencing presents the *PorductionMethods* for *Feature* and *PartFamily(s)* that consist of clabjects. The *Stages*, *Operations*, *FeatureProductionMethods* and *PartFamilyProductionMethods* depicted in figures 8.12 to figure 8.16 are not *individuals*. They are the clabjects that can instantiate the *individuals*.

Using the proposed set of core concepts, the method of capture and reasoning about Meta level knowledge, it has been shown that;

- A significant contribution has been made towards the capture and acquisition of Meta level knowledge.
- The Meta level *ProductionMethod* for Feature(s) and PartFamily(s) can effectively be captured and reasoned about.
- It has been made possible to reason about the manufacturability of *Stage(s)* in *Operation(s)* at Meta level.
- It has been made possible to reason about the accommodation of *MetaFeatureProductionMethod(s)* in *PartFamilyProductionMethod(s)*.

8.9 Case Study

This case study is conducted to show that the proposed ontology and methodology can be applied to design and production scenarios beyond the circumferential groove. A different feature i.e. WebProfile (for production) or Diaphragm (for design) is considered and an experiment similar to experiment 3 has been performed. The main objective of this case study is to further strengthen the research argument verifying the research objectives for a different feature.

8.9.1.1 The Capture of Production Knowledge

It was known from the industrial study that the WebProfile of the disc should be machined in a single pass to avoid a surface irregularity (as shown in figure 8.17). In

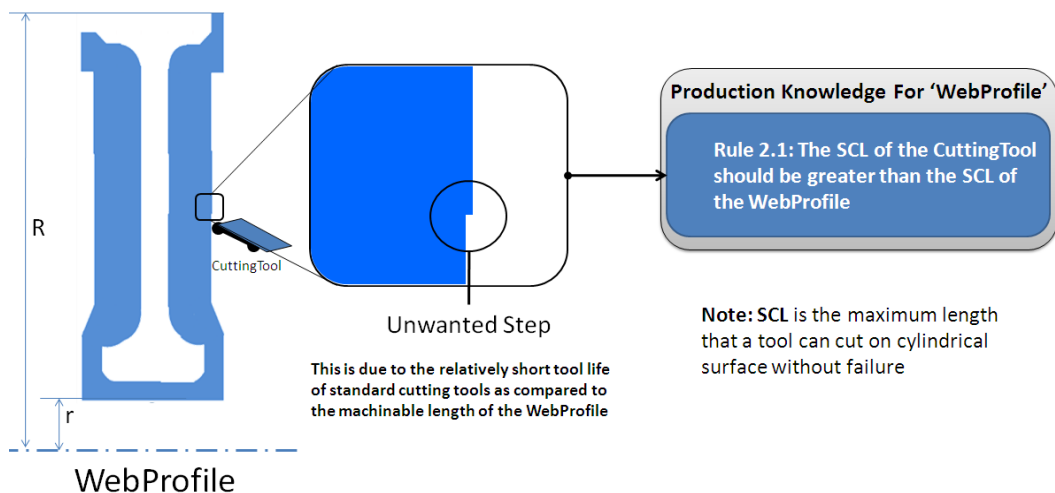


Figure 8.17: The knowledge to be captured for the Rim

order to machine the web profile in a single pass “The spiral cutting length of the CuttingTool should be greater than the spiral cutting length of the WebProfile”

The spiral cutting length (SCL) of cutting tools is the total length that a cutting tool can machine on a cylindrical or disc surface. SCL directly depends on the feed rates and are provided by the manufacturers of tools in relation to the feed rates. However, the

SCL of surfaces needs to be calculated. In order to capture the machine-ability of a WebProfile the SCL of WebProfile and cutting tool should be known.

In order to reason about the machine-ability of a WebProfile, the SCL of WebProfile has to be evaluated in relation to the SCL of *CuttingTool(s)* and the appropriate feedback should be given to the designer. The formalisation to capture the SCL of WebProfile is based on the following equation provided by Sandvik (2011);

$$SCL = \{(r + R)/0.31831\} * \{(R - r)/0.15\}$$

Where, 0.15 = assumed constant federate in mm/rev,
0.31831 = (1/π)

The relations and axioms are coded in KFL to formalise the values of r , R , $(r+R)$ and $(R-r)$ before writing the code for the SCL. The following relations and functions shown in table 8.4 are required to formally capture the SCL.

Table 8.4: The relations and function for capturing Rim's production knowledge

Relation	Corresponding variables
<i>hasParameter</i>	For associating R and r to WebProfile
<i>hasRminusrValue</i>	To infer $(R-r)$
<i>hasRminusrbyfeed</i>	To infer $(R-r)/0.15$
<i>hasRplusrValue</i>	To infer $(R+r)$
<i>hasRplusrValuebyconstant</i>	To infer $(R+r)/0.3183$
<i>cuttingToolhasSC</i>	To capture the SCL of the CuttingSpeed
<i>webProfilehasSCL</i>	To infer the SCL of WebProfile
<i>gteNum</i>	For Relating Greater than equal to

The knowledge about the manufacturing of WebProfile is finally captured in the following axiom.

```
:WebProfile's Production Axiom 1
;Axiom for Predicting machine-ability of WebProfile
(=> (and (WebProfile ?wp)
(WebProfilehasSCL ?wp (MCCO.mm ?wpSCL)))
(exists (?ct ?ctSCL)
(and (MCCO.CuttingTool ?ct)
(cuttingToolhasSCL ?ct (MCCO.mm ?ctSCL))
(gteNum ?ctSCL ?wpSCL))))
:IC hard "The available tools cannot machine the asserted WebProfile in a single pass"
```

The message of IC states the logic captured in the axiom in simple words. The above axiom will disallow the assertion of any WebProfile whose SCL is greater than the SCL of the available *CuttingTool(s)*. Now that the knowledge has been captured it can be shared with the relevant design feature through the knowledge sharing route.

8.9.1.2 Establishing the Route to Knowledge Sharing

Diaphragm is one such *Feature* from the AeroEngineDiscDesignOntology. It is understood that the maximum SCL of cutting tools at the standard speed of 50 m/min is 1,375,000 mm. Consider a case where the designer changes length of *Diaphragm* such that it changes Radii of the disc to $R=262$ mm and $r=50$ mm. The production consequences of this design change can be known if the *DiscProductionFeature(s)*

relevant to the 'Diaphragm1' is identified. Based on the same lines as that of experiment 3 (section 8.7.1.2), the *DiscProductionFeature* relevant to 'Diaphragm1' is found to be 'Web1' as shown in figure 8.18. The results of this query has provided a route to link the 'Diaphragm1' to its corresponding disc production feature 'Web1'

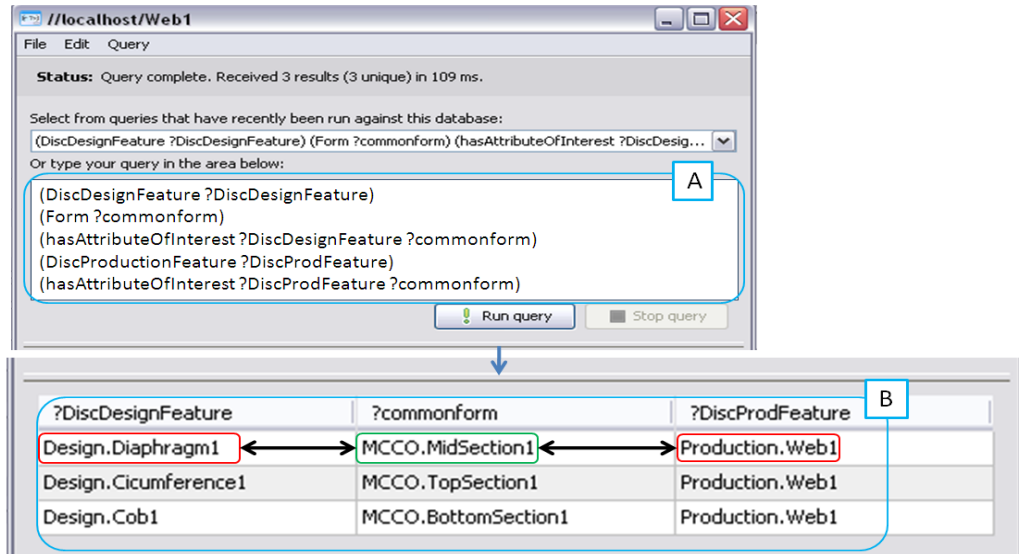


Figure 8.18: The query and its results to establish the route to knowledge sharing

8.9.1.3 Getting Feedback from Production into Design

The feedback from production into design is passed by asserting the designed values of R and r of Diaphragm1 in the XKS for 'Web1'. The results of the assertion of Rim1 constitute the feedback for 'Diaphragm1'. First assertion is made with R = 262mm and r = 50mm (figure 8.19).

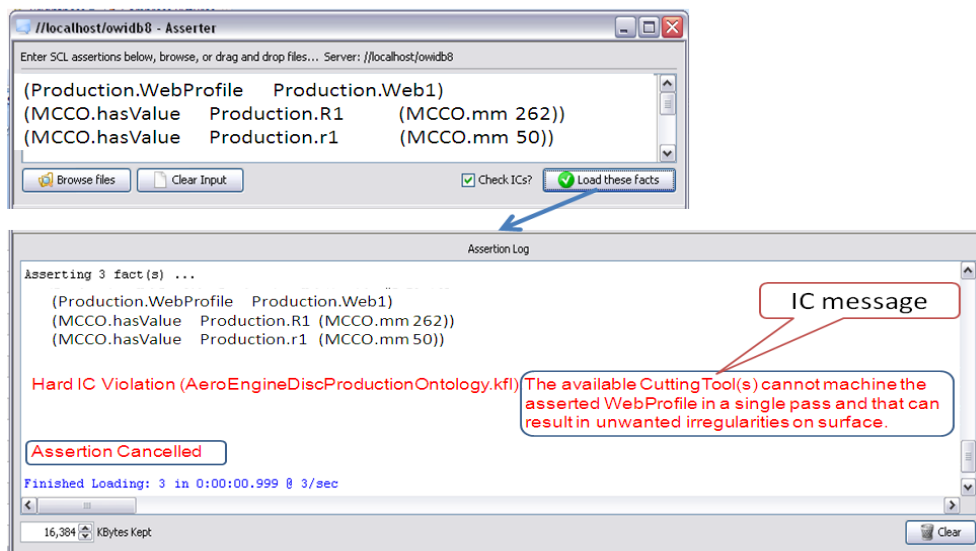


Figure 8.19: Assertion of a WebProfile fact Web1' with R = 262mm and r = 50mm

As shown in the figure 8.19, the assertion was cancelled as a result of the violation of ICs from the AeroEngineDiscProductionOntology. The messages of the violated ICs constitute the production knowledge feedback for the design.

This can be inferred from the IC message that dimensions of Diaphragm1 are beyond the production capacity. Therefore, a change in design is made and the value of R is changed from 262mm to 260mm and re-asserted in the XKS. The feedback is in the form of design being acceptable as shown by the non-violation of ICs and acceptance of facts in figure 8.20.

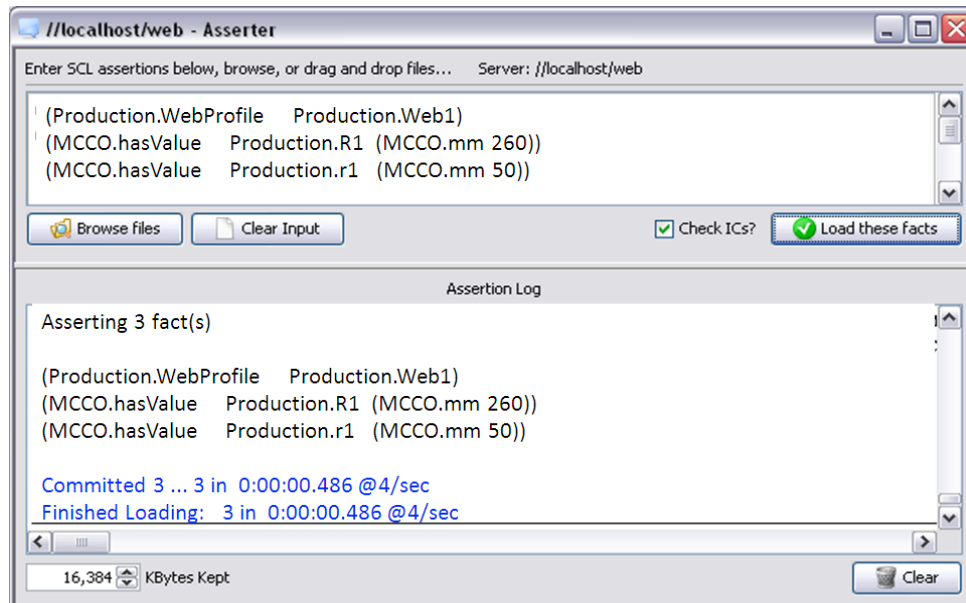


Figure 8.20: Assertion of a WebProfile fact Web1' with R = 260mm and r = 50mm

8.9.2 Discussion and Conclusions

An important point in this experiment is regarding the one to many relations between the design and production features. As shown in figure 8.18, the Web1 does not only map across to Diaphragm1 but it also maps to two other *DesignFeatures* namely 'Circumference1' and 'Cob1' through their common form. This shows that the forms of design and production features are not the same but they can have overlapping portions that can help to establish the link between design and production features.

The conclusions of this experiment are same as that of experiment 3. The only different is that the case study has tested the use of the MCCO beyond the CircumferentialGroove. This case study verifies that the MCCO is not just limited to the studied feature and has a potential to be widely applicable.

8.10 Summary

This chapter has experimentally verified various research aspects. The chapter has experimentally shown that the semantics of the concepts have been captured in the MCCO using formal logic. It has been established that the proposed specialisation levels in the MCCO enable capturing the varying depths of meaning of concepts from

generic to the domain specific concepts. It has been demonstrated the MCCO provides a commonly understood semantic base for developing the semantically sound application specific product design and production ontologies. The MCCO also provides a route to knowledge sharing by linking the application specific domains through the foundation provided by the MCCO.

9 Discussion, Conclusions and Future Research

9.1 Introduction

The research work reported in this thesis has explored the use of a heavyweight ontological approach to support knowledge sharing across multiple domains. It has been shown that the approach can support knowledge sharing across product design and production domains through the development and use of a heavyweight manufacturing core concepts ontology (MCCO) as a common semantic base to support specific design and production ontologies. This chapter provides a discussion, a set of conclusions and a guideline for future research work based on the research work reported in this thesis. The chapter is organised as follows.

A discussion on different aspects of the research work is presented in section 9.2. The discussion leads to a set of conclusions and recommendations for future research that are respectively reported in sections 9.3 and 9.4.

9.2 Discussion

The set of concepts and relations in the MCCO has been explored with a view to provide a semantic base to enable the capture and sharing of production (mainly machining) knowledge with product design. The research work reported in this thesis has been conducted to achieve the research objectives listed in section 1.3. In order to achieve those objectives, a number of research questions were raised in chapter 4. An appraisal of the research work against those research questions and how they lead to the achievement of research objectives is presented in the discussion section. A critical analysis of the reported research work is also presented in this section.

9.2.1 The MCCO as an Intermediate Set of Concepts

The following questions were raised in section 4.3 of chapter 4 regarding the requirement to have an intermediate set of concepts and relations between product design and production domains and the formalisation of their semantics.

Q1 (section 4.3): What are the intermediate or core set of manufacturing concepts that sit between the foundation and application specific product design and production domain ontologies?

Q4 (section 4.3): Can the semantics of the core concepts and relations be captured formally using heavyweight logic so that the knowledge system can

computationally understand the meanings of concepts and thus help identify and remove the ambiguities in their interpretations?

The answer to the question “Q1 (Section 4.3)” helps to achieve the research objectives 1 and 2 mentioned in section 1.3. The answer to the question “Q4 (Section 4.3)” helps to achieve the research objective 3 mentioned in section 1.3. Chapter 5 details the proposed answer to the above research questions. Following the proposed ontology development method a core set of manufacturing concepts and relations were proposed. The set of concept and relations identified in the MCCO are neither as generic as foundation ontologies and nor as specific as the application specific domain ontologies. However, this core set of concepts provides a common semantic base for knowledge sharing across application specific product design and production ontologies. A lightweight formalisation of these concepts and relations i.e. the MCCO was presented in section 5.4 of chapter 5. The proposed manufacturing ontology established the achievement of objectives 1 and 2 of this thesis.

The answer to the question “Q4 (Section 4.3)” helps to achieve the research objective 3 mentioned in section 1.3. Section 5.5 of chapter 5 addresses the question “Q4 (Section 4.3)” by presenting the formalisation of the semantics of the concepts in formal logic. This enables the system to understand the semantics of the concepts and respond accordingly. The experimental verification of the semantics of the concepts has been reported in section 8.5 of chapter. The experimental investigation showed that the system understood the semantics of the concepts and issued warnings and even cancelled the assertion of facts when the formal definitions were violated. This shows that the research objective 3 of the thesis has been successfully met.

The thesis has shown the potential value of a MCCO as a multi-domain knowledge sharing approach, focusing on providing machining knowledge back to design. However, for the MCCO to be adopted by the manufacturing industry and be validated by the end users, further exploration may be required in certain areas. For example, the MCCO focused on identifying the concepts that facilitate the production domain and within production the area of conventional machining. Even in the machining domain, the concepts identified are generic being at the core level. Therefore, extension of the MCCO is perhaps required to identify the set of concepts and relations that are more useful directly for the industry. For example the concepts like turning, milling, drilling, boring and similar other concepts. There are number of different ontologies that provide different hierarchies of manufacturing processes. Each industry may have its own preference on choosing or defining their own hierarchy of concepts.

However, research should be conducted to define a standard set of concepts that provide generally agreed formal definitions for manufacturing processes, manufacturing resources, manufacturing facilities and realised parts.

Similarly, the MCCO should not only support machining but also other areas within production like assembly, casting, rapid prototyping, forging and non conventional manufacturing. Therefore, the MCCO should be extended to provide core concepts and relations for other production areas. On the same lines, the MCCO should also be extended to act as a common semantic base across other product lifecycle domains like services, operations, maintenance and disposal.

9.2.2 Capturing the Varying Depths of Meaning of Concepts

Research objective 4 (section 1.3) was to enable the capture of varying depth of meaning of concepts. In order to achieve this objective, a research question was raised in section 4.4 of chapter 4.

Q1 (section 4.4): How can the varying depths of meaning of manufacturing concepts, from generic to the specific concepts, be effectively and formally captured?

The answer to this question should help to achieve the research objective 4 of this thesis. Sections 6.3 and 6.4 of chapter 6 report an approach of specialising concepts at different levels to enable the formal capture of varying depths of meaning of concepts. Three different levels of specialisations were proposed to capture the depths of meaning from the generic to the domain specific levels. The approach ensured consistency of semantics over the specialised concepts. This approach was experimentally verified in section 8.5 of chapter 8. This shows that the research objective 4 of the thesis has been successfully met.

It was identified that the core concepts can be generic to any domain, can be generic to any of the product lifecycle domains and can be generic to the product design and production domains. Therefore three levels of specialisations i.e. (1) generic core concepts level, (2) product lifecycle generic core concepts level (3a) design generic core concepts level (3b) production generic core concepts level were identified to capture the corresponding variations in the depths of meaning of concepts.

However, the variations in the depths of meaning of concepts can exist at even more detailed levels. For example, in production domain further levels can be explored to capture the core concepts generic to machining concepts, core concepts generic to casting and core concepts generic to forging. Similarly, the varying depths of

meanings and their corresponding levels of specialisations can be explored for other product lifecycle domains like operations, services, and disposal. Such research may lead to several different variations in depths of meaning of concepts and consequently different levels of specialisations. Therefore, the levels of specialisation of core concepts can be explored further in with respect to different levels details that the core concepts are aimed at providing.

9.2.3 Developing Application Specific Ontologies

Another objective i.e objective 5 of this thesis is to use the formally defined set of concepts and relations to support the development of semantically sound application specific product design and production ontologies. In this regard, the following research question was raised in section 4.3 of chapter 4.

Q2 (section 4.3): Can the core set of manufacturing concepts and relations support the development of application specific product design and production ontologies?

An approach to address this question is reported in section 6.5 of chapter 6, which showed that the set of concepts and relations in the MCCO could support the development of semantically sound application specific product design and production ontologies. The experimental verification of this has been presented in section 8.6 of chapter 8. That showed that the research objective 5 of the thesis has been successfully met. However, the application specific ontologies developed using the MCCO needs to be tested on a broader scale to show the broader applicability of the MCCO.

The application specific ontologies developed in this thesis mainly used the *Feature* concepts with other concepts from the MCCO being involved indirectly. The application specific ontologies developed are based on the examples from an aero engine industry. For the MCCO to have broader application and use, it should be exploited for its ability to support the development of application specific ontologies in other application area within the manufacturing like production and machining examples from the automotive industry and machining of moulds. Therefore, the MCCO can also be investigated for its ability to support the development of application specific ontologies in such different focus areas within the manufacturing domains. The MCCO should also be validated through different industrial implementations to show the application and effectiveness of the MCCO in the industrial environments

9.2.4 The Route to Knowledge Sharing between Design and Production

The objective no. 6 of the research was to provide a route for knowledge between the application specific domains. In this context, the following research question was raised in section 4.3 of chapter 4.

Q3 (section 4.3): Can these core manufacturing concepts and relations provide a route for knowledge sharing between product design and product domains?

The research question regarding the route to knowledge sharing i.e. Q3 (section 4.4) was addressed in section 6.6 of chapter 6, which showed that the MCCO can provide a route to link different product design and production ontologies. The experimental verification of this route has been reported in sections 8.7.1.2 and 8.9.1.2 of chapter 8.

The route to knowledge sharing was established through the *Feature* concepts where different features are related through the overlapping portions of their forms. Therefore, the features should be defined to have one or more forms that can be used to relate different design and production features (defined in machining context). However, it is possible that the designers and production engineers do not specify any forms and they just produce and work with parametric models of parts. In such a case, the route to knowledge sharing will need to be explored at the parametric level. Therefore, this needs to be explored and it can be an area for future research work.

9.2.5 Capturing and Reasoning about Meta and Higher Levels of Knowledge

A requirement to capture and reason about the production knowledge at the Meta level knowledge was established in section 4.5.2 of chapter 4. In the same section, the following research questions were raised.

Q1 (section 4.5): What are the required concepts and relations that can support the capture and reasoning about Meta level manufacturing knowledge?

Q2 (section 4.5): How can the Meta level manufacturing knowledge be formally represented, captured, and reasoned about using those *concepts*?

Sections 7.3 and 7.4 of chapter 7 present the formalisation of the core set of concepts and relations that enable the knowledge capture and reasoning at the *Meta level knowledge*. The experimental verification of the ability of these concepts and relations to enable the capture of *Meta level production methods* and support reasoning over them is presented in section 8.8 of chapter 8.

The present levels addressed in this research are *MetaMetaLevel* --> *MetaLevel* --> *IndividualLevel*, where the *MetaMeta level* and the *Meta level* contain concepts and the *Individual level* contains the individuals. The concepts i.e. powertypes at the *MetaMeta level* enable knowledge capture and reasoning over the clabjects which are at the *Meta level*.

Similarly, the concepts i.e. clabjects at the *Meta level* support the capture and reasoning at the *Individual level knowledge*. The number of levels of abstractions of knowledge can be as many as required. However, in this thesis a requirement to capture and reason about the *Meta level knowledge* was identified. This requirement was satisfied by introducing the concepts at *MetaMeta level* known as powertypes. If a requirement to capture and reason about the *MetaMeta level knowledge* was established, than that may be satisfied by introducing the concepts and relation at the *MetaMetaMeta level*. The same approach could be applied to even more abstract levels depending upon the levels of abstraction at which it is required to capture and reason about knowledge. Therefore, this approach provides a method to capture and reason about knowledge at multiple levels of abstractions. This is an exciting aspects of this approach because it is not possible to capture and reason about the *Meta level knowledge* with traditional knowledge modelling approaches.

However, in order to make use of this approach it is required that the *Meta level knowledge* to be provided as the extension of the ontology and cannot be asserted like individual facts. This is because the *Meta level knowledge* consists of clabjects, which are concepts and not individuals. Addition or removal of clabjects affects the structure of the ontology. This could become an issue if there were regular changes in the *Meta level knowledge* because this will require changes in the structure of the ontology. The present knowledge and database management systems are not equipped with the ability to handle multiple levels of abstraction of knowledge. Therefore, as the complexity of the knowledge environment is extended beyond production methods and at different levels of knowledge abstractions, different issues in modelling of knowledge, and usage and implementation of the approach may arise.

9.2.6 Other Aspects of the Research

9.2.6.1 Extension of the MCCO

During the development of the application specific ontologies in sections 6.5 and 8.6 of chapters 6 and 8 respectively, the concepts like *TurningFeature*, *MillingFeature*, and *DrillingFeature* were shown as being part of the application specific ontologies. There can be an argument that concepts like these can be part of the MCCO because they are applicable to a wide range of if not all manufacturing organisations. Similar

arguments can apply to extend the ontology for the different *ManufacturingProcess* concepts like *TurningProcess*, *MillingProcess* and similar argument applies to defining concepts for *Operations*, *Stages* and so on.

However, a boundary line needs to be drawn to separate the core concepts from application specific concepts. Although concepts like *TurningFeature* and *MillingOperation* can be applicable to most manufacturing organisation, they have not been included in the MCCO. This is because the MCCO provides the semantics and structure for *PartFamily*, *Feature*, *ProductionMethod*, *Operation*, *Stages* etc. These concepts provide the basic understanding and semantics required to develop application specific ontologies without constraining the application domain to use the specific concepts like *TurningFeature*, *MillingFeature* etc. The more specific the ontology is, the more difficult it is for the application specific domains to agree on that ontology. Therefore, the defined generic nature of the MCCO makes it suitable as a core ontology for different application specific domains and simultaneously provides an effective route for knowledge sharing across them.

The MCCO can however, still be extended in these different directions that can be marked as extensions of the MCCO. In fact, these extensions have been developed for some concepts like *ManufacturingProcess* in this thesis, which can be referred to in section A2.6 of appendix A2.

9.2.6.2 Broader Effectiveness of the MCCO

For the MCCO to be effective on a broader scale and industrially exploitable, this should be used for application beyond the applications presented in this thesis. This means that the MCCO should be useful for knowledge sharing beyond the design and production domains of an aero engine disc. The MCCO has been proposed to support knowledge sharing across a wider range of design and production applications. In this regard, it is pleasing to report that the MCCO has successfully been implemented and used in two projects i.e. 'Strategic Affordable Manufacturing in the UK with Leading Environmental Technology (SAMULET)' projects SAMULET-5.6.1 and SAMULET-3.7.3.

Elements of the MCCO have been used and extended in SAMULET-5.6.1 MCCO to support the development of a manufacturing knowledge maintenance system. This system mainly made use of the *Feature* and *PartFamily* concepts and used a lightweight representation of these concepts. This system defined methods for the capture and maintenance of manufacturing knowledge such that it can be shared with the product design and production disciplines.

MCCO has also been utilised in SAMULET 3.7.3 during the development of a Life Cycle Knowledge Desktop (LCKD). An OWL based version of the MCCO was developed to in accordance with the requirements of the LCKD. The LCKD offered the ability to access multiple sources of information by utilizing, rather than recreating, existing knowledge that is spread across the company network. The LCKD comprises the development and application of effective document search technologies across the lifecycle of aero engine product development. MCCO has particularly supported the development and delivery of a high-level manufacturing ontology with additional extended local ontologies to capture the best practice organization of manufacturing information in the context of LCKD.

The MCCO has successfully met the IMKS project objectives by providing a high-level library of manufacturing core concepts and relation that can be extended and specialized to develop specific domain ontologies. A prototype PLM support system has been built by the IMKS research team based on the MCCO ontology.

The successful use of the MCCO in the above-mentioned projects establishes the MCCO as an effective manufacturing ontology that has the potential to act as reference manufacturing ontology for several product lifecycle disciplines.

However, the MCCO itself has not been directly and fully validated by the industry. For the MCCO to be validated and adopted as part of knowledge sharing system the research work needs to be extended in the following direction. Firstly, as mentioned in section 9.2.1, the extensions to the MCCO should be considered to include a broader range of machining concepts, concepts and relations for other production areas and for other product lifecycle domains.

In this thesis, the design core concepts are mainly belonging to the *Feature* and *PartFamily* categories of concepts. The MCCO can also be extended and explored further for a more comprehensive set of core concepts and relations for the product design domain. For example, in this thesis all the different design requirements i.e. functional requirements, assembly requirements, thermal stress conditions and working stress conditions have all been merged into the design function. Although this is correct at a generic level but in practice, different specialists are there for conducting detailed stress analysis, thermal analysis, assembly, and other functional requirements. Each one of these design areas dictate the design of parts and features in their own specific ways. On detailed investigation, it is possible that the features of interest for a stress analyst are different from the features of interest to a functional design engineer. In such a case, what would be the chances of successful

knowledge sharing between different design domain and how can those be enhanced. Therefore, research work needs to be conducted to define the core set of concepts that can facilitate the different design areas, support knowledge between different design areas and between design and production.

For the MCCO to be implemented as a software tool i.e. a computational knowledge sharing system, the MCCO should be explored further in the areas discussed above, a broader and comprehensive manufacturing ontology covering all the different relevant areas useful for the manufacturing industry should be formally defined. The ontology and the knowledge base system should be developed into a tool by developing a front end knowledge management interface. The knowledge sharing tool can also be developed by integrating ontology and the KB into a product lifecycle management system like Siemens PLM.

The research work reported in this thesis is limited within the scope of the thesis and cannot cover the broad range of research issues and areas. However, an understanding and an approach has provided that can be followed on to develop a fully functional and industrially exploitable knowledge sharing tool. A contribution to knowledge has been made in terms of the achievement of the research aims and objectives. This contribution can be summed up as a verification of the research hypothesis as;

“An ontology (The MCCO) of a comprehensive set of core manufacturing concepts defined in formal logic can support knowledge sharing across product design and production domains by providing a verifiable semantic base.”

9.3 Conclusions

The following set of conclusions has been drawn from the discussion.

- It has been shown that the heavyweight core concepts ontology and approach provide a basis for knowledge sharing from a production domain into a design domain.
- It has been shown that through the adapted ontology development methodology, a lightweight manufacturing core concepts ontology that captures the key concepts and relations needed to support the capture and sharing of production knowledge into product design can be developed.

- It has been shown that the semantics of concepts have been successfully captured in formal logic, which enabled the computer systems to understand the semantic of concepts and respond in accordance with the formal definition of concepts.
- It has been shown that the approach to specialise concepts at the three proposed levels enables the system to formally capture the varying depths of meaning of concepts. This approach also ensures the formal semantic consistency and inheritance over the specialised concepts.
- It has been shown that MCCO has the ability to support the development of semantically sound application specific product design and production ontologies.
- It has been shown that the MCCO provides a route to knowledge sharing between product design and production domains. The *Feature* concepts defined in a *PartFamily* context are the key to this knowledge sharing route.
- It has been shown that it is possible to capture and reason about the production methods at the *Meta level Knowledge*. This was made possible by proposing an approach that involved the use of clajects and powertypes. This is a step improvement in manufacturing knowledge engineering fields.

9.4 Future Research

The discussion also highlighted the need to conduct the research work to take further the research work conducted in this thesis. Recommendations for future research work are listed below.

1. To make the ontology directly more applicable in an industrial environment, research work should be conducted to define the core set of machining concepts that are generally e.g. core concepts for machining processes, machining operations, fixtures and types of machines.
2. Within the production domain, the set of concepts and relation can be explored for other production processes such as the assembly, casting and non-conventional machining by defining the core concepts suitable for these domains.
3. With the product lifecycle domain, the set of concepts can be explored to provide a semantic base for the domains like Operations, Maintenance & Services, and Disposal. Therefore, the MCCO can be expanded as a reference ontology for manufacturing to increase its applicability outside of production and into other manufacturing domains.

4. The levels of specialisations defined to capture the varying depths of meaning of concepts need to be explored further for other possible variations in the depths of meaning of concepts. The number of levels of specialisation may increase from three.
5. A more detailed level of interoperability between design and production features can be explored at a parametric level. This is because the current approach assumes that the features are defined with form(s). Although, interoperability has been explored through the overlapping forms between design and production, the identification of overlapping forms is based on the matching of portions of forms of design and production features. A more detailed and comprehensive way of identifying the overlapping forms can be explored by matching the forms through their parameters.
6. The use of clabjects and powertypes may be explored to capture and reason about even higher levels of abstraction of knowledge.
7. Research work can be directed to formalise the useful concepts from various standards in common logic. For example, the text based non formal definitions of concepts can be formalised for standard like ISO-STEP-10303-Ap224, AP-1 etc, ISO-18629-PSL, ISO-15531-MANDATE, ISO-13584-PLib etc to support coherence, uniformity and formal consistency across standards.
8. A research direction for future is also to exploit the manufacturing core concepts ontology to support interoperability between different ontologies from the same domain. For example, the exploration of the MCCO to support interoperability between different production domain ontologies and the interoperability between different design domain ontologies.
9. Features in the design domain can be different with respect to different areas of with the design domain. Therefore, research work should be conducted to define a more comprehensive set of core concepts that can facilitate the modelling of different design areas like stress analysis, thermal analysis and function based modelling.
10. The proposed approach can be applied to support interoperability across domains other than those belonging to manufacturing. For example, the heavyweight core concepts ontological approach can be investigated for its usefulness in different business domains, medical sciences and civil and buildings engineering.

Publications

Journal Papers

1. N. Chungoora, G. Gunendran, R.I.M. Young, **Z. Usman**, N.A. Anjum, C. Palmer, J.A. Harding, K. Case, and A.F. Cutting-Decelle. **Extending product lifecycle management for manufacturing knowledge sharing**. [prepared for submission]
2. N. Chungoora, R.I.M. Young, G. Gunendran, C. Palmer, **Z. Usman**, N.A. Anjum, A.F. Cutting-Decelle, J.A. Harding and K. Case. **A model-driven ontology approach for manufacturing system interoperability and knowledge sharing**. Computers in Industry. [submitted for review 15 December 2011]
3. N. Chungoora, A.-F. Cutting-Decelle, R.I.M. Young, G. Gunendran, **Z. Usman**, J.A. Harding, K. Case, **Towards the ontology-based consolidation of production-centric standards**, International Journal of Production Research, DOI:10.1080/00207543.2011.627885.
4. C. Palmer, A.G. Gunendran, R.I.M. Young, N. Chungoora, **Z. Usman**, K. Case and J.A. Harding **Exploiting UML as a Preliminary Design Tool for Common Logic-based Ontologies**, International Journal of Computer Integrated Manufacturing, [submitted for review October 2011]

Conference and Workshop Papers

5. N. Chungoora, R.I.M. Young, G. Gunendran, **Z. Usman**, N.A. Anjum, C. Palmer, J.A. Harding, K. Case, and A.F. Cutting-Decelle. 2012. **A Model Driven Ontology Based Approach for Manufacturing Knowledge Sharing in PLM**. In: Workshop on Interoperability for Enterprise Software and Applications (I-ESA). Valencia, Spain. March 20-21
6. **Z. Usman**, R.I.M. Young, N. Chungoora, K. Case, C. Palmer, J.A. Harding, **A manufacturing core concepts ontology for product lifecycle interoperability**, International Working conference on enterprise interoperability, Stockholm Sweden, March 22-23, 2011.

7. R.I.M. Young, N. Chungoora, **Z. Usman**, N.A. Anjum, G. Gunendran, C. Palmer, J.A. Harding, K Case and A-F Cutting-Decelle, **An Exploration of Foundation Ontologies and Verification Methods for Manufacturing Knowledge Sharing**, Workshop on Interoperability for Enterprise Software and Applications (I-ESA Workshop, 2010), Coventry, UK, April, 13, 2010
8. **Z. Usman**, R.I.M. Young, K. Case, J.A. Harding, **A Manufacturing Foundation Ontology for Product Life Cycle Interoperability**, The international conference on Interoperability for Enterprise Software and Applications (I-ESA 2010), Coventry, UK, April, 14-15, 2010.
9. Robert I.M. Young, N. Chungoora, **Z Usman**, N Anjum, G. Gunendran, C. Palmer, J Harding, K Case and A-F Cutting-Decelle, **Reference ontologies for manufacturing based ecosystems**, International Working conference on enterprise interoperability, Stockholm Sweden, March 22-23, 2011.

References

- A.F. Cutting-Decelle, R.I.M. Young, J.J. Michel, R. Grangel, J. Le Cardinal and J.P. Bourey. 2007. " ISO 15531 MANDATE: A Product-process-resource based Approach for Managing Modularity in Production Management." *Concurrent Engineering* 15 217-235.
- Abdul-Ghafour, Samer, Parisa Ghodous, Behzad Shariat, Eliane Perna, Daniel D. Frey, Shuichi Fukuda and Georg Rock. 2011. "Ontology Development for the Integration of CAD Models in a Collaborative Environment Improving Complex Systems Today." Springer London.
- Abramovici, Michael and Olaf C. Sieg. 2002. "Status and development trends of product lifecycle management systems." In IPPD conference. Wroclaw.
- Anjum, N.A., Harding, J.A., Young, R.I.M. and Case, K. . 2010. "Mediation of foundation ontology based knowledge sources. *Computers in Industry*. [submitted for review 24 September 2010]."
- Anjum, N.A., Harding, J.A., Young, R.I.M. and Case, K. 2011. "Manufacturability verification through feature based ontological product models. ." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*. [submitted for review 4 March 2011].
- Anjum, N.A. 2011. "Verification of knowledge shared across design and manufacturing using a foundation ontology." Loughborough University.
- Anjum, N.A., J.A. Harding and R.I.M. Young. 2010. "Cross domain knowledge verification: Verifying knowledge in foundation based domain ontologies." In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)* Valencia, Spain.
- Anjum, N.A., J.A. Harding and R.I.M. Young. 2011. "Shape feature based ontological engineering product models." In *3rd International IFIP Working Conference on Enterprise Interoperability (IWEI)*. . Stockholm, Sweden. .
- Anjum, N.A., J.A. Harding, R.I.M. Young and K. Case. 2010. "Gap analysis of ontology mapping tools and techniques." In *Enterprise interoperability IV: Making the Internet of the future for the future of enterprise – Proceedings of the 6 th International Conference on Interoperability for Enterprise Software and Applications (I-ESA)*. , eds. K. Popplewell, J.A. Harding, R. Poler and R Chalmers. Coventry, UK.
- Antoniou, Grigoris. and Frank Van Harmelen. 2008. *A Semantic Web Primer 2nd Edition*. Cambridge: MIT Press.
- Arthur, Jeffrey B. and Christopher L. Huntley. 2005. "Ramping up the Organizational

- Learning Curve: Assessing the Impact of Deliberate Learning on Organizational Performance under Gainsharing." *The Academy of Management Journal* 48(6):1159-1170.
- Arturo, Molina, Gutierrez. 1999. "A manufacturing model representation of a flexible manufacturing facility." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 213(3):225-246.
- ATHENA. 2004. " D.A.1.1.1. State of the art of enterprise modeling techniques and technologies to support enterprise interoperability (www.athena-ip.org)."
- ATHENA. 2007. "D.A4.2: Specification of Interoperability Framework and Profiles, Guidelines and Best Practices", . " ATHENA Integrated Project, Deliverable D.A4.2,.
- Atkinson, C. and T. Kühne. 2000. "Meta-Level Independent Modelling " In *International Workshop on Model Engineering at 14th European Conference on Object-Oriented Programming*. Sophia Antipolis and Cannes, France.
- Aussenac-Gilles, Nathalie, Brigitte Biébow and Sylvie Szulman. 2000. "Revisiting Ontology Design: A Method Based on Corpus Analysis." In *Knowledge Engineering and Knowledge Management Methods, Models, and Tools (EKAW2000)*. Berlin Germany: Springer.
- Babcock, Pamela. 2004. "Lessons learned lead to new ideas about sharing information-in Shedding Light on Knowledge Management." In *HR Magazine*.
- Baxter, David, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane and Shilpa Dani. 2008. "A framework to integrate design knowledge reuse and requirements management in engineering design." *Robot. Comput.-Integr. Manuf.* 24(4):585-593.
- Bechhofer, Sean, Manfred Hauswirth, Jörg Hoffmann, Manolis Koubarakis, Ernesto Jiménez-Ruiz, Bernardo Grau, Ulrike Sattler, Thomas Schneider and Rafael Berlanga. 2008. "Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support." In *The Semantic Web: Research and Applications*: Springer Berlin / Heidelberg.
- Beernaert, Dirk , Jonathan Borg, Teresa de Martino, Paul Drews, Peter Friess, Georg Kelm, Ron J. Patton, Rolf Riemenschneider, Gisèle Roesems-Kerremans, Asbjorn Rolstadas, Jorge Santos, Alejandro Sanchez-Gruoso, KlausDieter Thoben, Marco Taisch and John van Meurs. 2005. "ICT for Manufacturing, Report of Meeting with Group of Representatives of Five Expert Panels." Brussels.
- Benjamin, Perakath, Mukul Patki and Richard Mayer. 2006. "Using ontologies for simulation modeling." In *Proceedings of the 38th conference on Winter simulation*. Monterey, California: Winter Simulation Conference.
- Berre, A., B. Elvester, N. Figay, C. Guglielmina, S. Johnsen, D. Karlsen, T. Knothe and S. Lippe. 2007. "The ATHENA Interoperability Framework." In *3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007)*. Madeira, Portugal: Springer.

- Bless, Patrick, N. , Diego Klabjan and Soo Chang, Y. 2008. "Heuristics for automated knowledge source integration and service composition." *Computers and Operations Research* 35(4):1292-1314.
- Blomqvist, Eva and Annika Öhgren. 2008. "Constructing an enterprise ontology for an automotive supplier." *Engineering Applications of Artificial Intelligence* 21(3):386-397.
- Bloodsworth, Peter and Sue Greenwood. 2005. "COSMOA an ontology-centric multi-agent system for co-ordinating medical responses to large-scale disasters." *AI Commun.* 18(3):229-240.
- Bloodsworth, Peter, Sue Greenwood and John Nealon. 2004. "COSMOA: An Ontology-Centric Multi-Agent System For Coordinating Medical Responses To Large-Scale Disasters." In *Proceedings of ECAI Workshop Agents Applied in Healthcare*. Valencia, Spain.
- Bombardier, Vincent, Cyril Mazaud, Pascal Lhoste, Rapha and I Vogrig, Vogrig. 2007. "Contribution of fuzzy reasoning method to knowledge integration in a defect recognition system." *Comput. Ind.* 58(4):355-366.
- Borgo, Stefano and Paulo Leitão. 2007. "Foundations for a core ontology of manufacturing." Bragança, Portugal.
- Borgo, Stefano and Paulo Leito. 2004. *The Role of Foundational Ontologies in Manufacturing Domain Applications*.
- Borst, W., N., . 1997. "Construction of Engineering Ontologies for Knowledge Sharing and Reuse." In *Centre for Telematica and Information Technology: University of Twente*.
- Bourey, J., P. 2007. "Model Driven Interoperability and Service-Oriented Architecture." In *IMS Seminar*-accessed online at <ftp://ftp.cordis.europa.eu/pub/ims/docs/2-6-bourey.pdf> on 17/06/2011. Zürich: InterOp V-Lab.
- Bourey, Jean-Pierre, Reyes Grangel, Guy Doumeingts and Arne J. Berre. 2007. "Report on Model Driven Interoperability. ." In *Technical Report: INTEROP, 2007*. http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables.
- Bradfield, D. J. and J. X. Gao. 2007. "A methodology to facilitate knowledge sharing in the new product development process." Taylor & Francis.
- Briefing-paper. 2008. "Identifier interoperability." Digital perspective Europe.
- Brunnermeier, S.B. and S.A. Martin. 1999. "Interoperability Cost Analysis of the U.S. Automotive Supply Chain." National Institute of Standards and Technology, U.S.A.
- Bussler, Christoph, Val Tannen, Irini Fundulaki, Philip Tan, Stuart Madnick and Kian-Lee Tan. 2005. "Context Mediation in the Semantic Web: Handling OWL Ontology and Data Disparity Through Context Interchange." In *Semantic Web and Databases: Springer Berlin / Heidelberg*.

- Chang, Xiaomeng, Rahul Rai and Janis Terpenney. 2010. "Development and Utilization of Ontologies in Design for Manufacturing." *Journal of Mechanical Design* 132(2):021009.
- Chang, Xiaomeng, Asli Sahin and Janis Terpenney. 2008. "An ontology-based support for product conceptual design." *Robotics and Computer-Integrated Manufacturing* 24(6):755-762.
- Chen, D., Knothe, T. and Zelm, M. 2004. "ATHENA integrated project and the mapping to International Standard ISO 15704. ." In *Proceedings of the International Conference on Enterprise Integration Modelling Technology*. Valencia, Spain.
- Chen, David, Guy Doumeingts and Francoise Vernadat. 2008. "Architectures for enterprise integration and interoperability:Past, present and future." *Computers in Industry* 59:647-659.
- Chen, Yuh-Jen, Yuh-Min Chen and Hui-Chuan Chu. 2009. "Development of a mechanism for ontology-based product lifecycle knowledge integration." *Expert Syst. Appl.* 36(2):2759-2779.
- Chen, Yuh-Min and Ching-Ling Wei. 1997. "Computer-aided feature-based design for net shape manufacturing." *Computer Integrated Manufacturing Systems* 10(2):147-164.
- Chungoora, Nitishal. 2010. "A framework to support semantic interoperability in product design and manufacture." In *Wolfson School of Mechanical & Manufacturing Engineering*. Loughborough: Loughborough University.
- Chungoora, N., Young, R.I.M., Gunendran, A.G., Palmer, C., Usman, Z., Anjum, N.A., Cutting-Decelle, A.-F., Harding, J.A., and Case, K. . 2011. "A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. ." *Computers in Industry*. [submitted for review 15 December 2011].
- Chungoora, N., Gunendran, G., Young, R.I.M., Usman, Z., Anjum, N.A, Palmer, C., Harding, J.A, Case, K. and Cutting-Decelle, A.-F. . 2012. "Extending product lifecycle management for manufacturing knowledge sharing. ." Submitted for Review in *International Journal of Production Research*.
- Chungoora, Nitishal, Osiris Canciglieri and R. I. M. Young. 2010. "Towards expressive ontology-based approaches to manufacturing knowledge representation and sharing." Taylor & Francis.
- Chungoora, N. and Bob Young. 2008. "Ontology Mapping to Support Semantic Interoperability in Product Design and Manufacture." In *International Workshop on the Model Driven Interoperability for Sustainable Information Systems*. Montpellier, France.
- Chungoora, Nitishal and Robert Ian Marr Young. 2010. "The configuration of design and manufacture knowledge models from a heavyweight ontological foundation." *International Journal of Production Research*.

- Chungoora, N. and R. I. M. Young. 2011. "An Ontological Approach to Consolidate Standards-Based Concepts in Production Engineering." *Semantic Computing (ICSC)*, 2011 Fifth IEEE International Conference on:514-521.
- Chungoora, Tish and Robert I. M. Young. 2008. "Semantic Interoperability Requirements for Manufacturing Knowledge Sharing." In *Interoperability for Enterprise Systems and Applications: I-ESA'08*.
- Cimiano, P., A. Eberhart, P. Hitzler, D. Oberle, S. Staab and R. Studer. 2004. "The SmartWeb foundational ontology-SmartWeb Project Report."
- CIMOSA-Association. 1996. "CIMOSA—Open System Architecture for CIM." Technical Baseline, Version 3.2.
- Clint, Chadwick and Dabu Adina. 2009. "Human Resources, Human Resource Management, and the Competitive Advantage of Firms: Toward a More Comprehensive Model of Causal Linkages." *INFORMS*.
- Cochrane, Sean, Robert Young, Keith Case, Jennifer Harding, James Gao, Shilpa Dani and David Baxter. 2008. "Knowledge reuse in manufacturability analysis." *Robotics and Computer-Integrated Manufacturing* 24(4):508-513.
- Cochrane, Sean, R.I.M. Young, Keith Case, Jenny A. Harding, James Gao, Shilpa Dani and David Baxter. 2009. "Manufacturing knowledge verification in design support systems." *International Journal of Production Research* 47(12):3179-3204.
- Cohen, W., P. Ravikumar and S. Fienberg. 2003. "A comparison of string distance metrics for name-matching tasks." In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*. Acapulco, Mexico.
- Collins, C. J. and K. G. Smith. 2006. "Knowledge exchange and combination: the role of human resource practices in the performance of high-technology firms." *Academy of Management Journal* 49(3):544-560.
- Corcho, Oscar, Mariano Fernandez-Lopez and Asuncion Gomez-Perez. 2003. "Methodologies, tools and languages for building ontologies: where is their meeting point?" Elsevier Science Publishers B. V.
- Correa da Silva, Flavio S., Wamberto W. Vasconcelos, David S. Robertson, Virginia Brilhante, Ana C. V. de Melo, Marcelo Finger and Jaume Agustí. 2002. "On the insufficiency of ontologies: problems in knowledge sharing and alternative solutions." In *Knowledge-Based Systems*.
- Court, A. W. 1998. "Issues for integrating knowledge in new product development: reflections from an empirical study." *Knowledge-Based Systems* 11(7-8):391-398.
- Cummings, Jonathon, N. 2004. "Work Groups, Structural Diversity, and Knowledge Sharing in a Global Organization." *Management Science* 50(3):352-364.

- Cutting, A. F., Decelle and J. J. Michel. 2003. "ISO 15531 MANDATE; a standardised data model for manufacturing management." Inderscience Publishers.
- Cutting-Decelle, A.F., J.-P. Bourey, R. Grangel and R.I.M. Young. 2006. "Ontology based communications through model driven tools - the MDA approach feasibility of the approach in urban engineering projects " In COST C21 1st Workshop on Ontologies for Urban Development – Interfacing Urban Information Systems. Geneva, Switzerland.
- Dartigues, Christel, Parisa Ghodous, Michael Gruninger, Denis Pallez and Ram Sriram. 2007. "CAD/CAPP Integration using Feature Ontology." *Concurrent Engineering*:237-249.
- Das, B. , A. F. Cutting-Decelle, R. I. M. Young , K. Case, S. Rahimifard, C. J. Anumba and N. Bouchlaghem. 2007. "Towards the understanding of the requirements of a communication language to support process interoperation in cross-disciplinary supply chains." *International Journal of Computer Integrated Manufacturing* 20(4):396-410.
- Davenport, Thomas H., W. De-Long and Micheal C. Beers. 2000. "Successful Knowledge Management Projects." In *The Knowledge Management YearBook 1999-2000*: Elsevier.
- Deshayes, Laurent, Sebti Foufou, Michael Gruninger, S. Tichkiewitch, M. Tollenaere and P. Ray. 2007. "An Ontology Architecture for Standards Integration and Conformance in Manufacturing Advances in Integrated Design and Manufacturing in Mechanical Engineering II." Springer Netherlands.
- Didonet del Fabro, M. , P. Albert, J. Bézevin and F. Jouault. 2008. "Industrial-strength rule interoperability using model driven engineering. Research Report Version 1." France: Institut de Recherche en Informatique et en Automatique, Rennes.
- Dong, Baoli, Guoning Qi, Xinjian Gu and Xiuting Wei. 2008. "Web service-oriented manufacturing resource applications for networked product development." *Advanced Engineering Informatics* 22(3):282-295.
- Doumeingts, G. and D. Chen. 2003. "Interoperability of Enterprise Applications and Software – An European IST Thematic Network Project:." In *IDEAS, Challenges Conference*. Bologna, Italy.
- DTI. 2005. "A Practical Guide to Cluster Development." In Department of Trade and Industry and the English RDAs Ecotec Research & Consulting.
- Elvesæter, Brian, Axel Hahn, Arne-Jørgen Berre and Tor Neple. 2006. "Towards an Interoperability Framework for Model-Driven Development of Software Systems." In *Interoperability of Enterprise Software and Applications*, eds. Dimitri Konstantas, Jean-Paul Bourrières, Michel Léonard and Nacer Boudjlida: Springer London.
- ESPRIT-Consortium-AMICE. 1993. "CIMOSA—Open Systems Architecture for CIM, 2nd revised and extended editionResearch Report, ." ESPRIT Project 688/5288, Springer-

Verlag.

Euzenat, J. 1996. "Corporative memory through cooperative creation of knowledge bases and hyper-documents " In 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96). Banff.

Fadel, Fadi George, Mark S. Fox and Michael Gruninger. 1994. "A generic enterprise resource ontology." In Proceedings of the third IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises Morgantown, West Virginia (WET ICE '94).

Fellbaum, Christiane. 1998. "WordNet: An Electronic Lexical Database." Cambridge, MA: MIT Press.

Feng, Shaw C. and Eugene Y. Song. 2003. "A manufacturing process information model for design and process planning integration." Journal of Manufacturing Systems 22(1):1-15.

Fenves, Steven J. , Sebtí Foufou, Conrad Bock and Ram D. Sriram. 2006. "CPM: A Core Model for Product Data."

Fernández-López, Mariano. 1999. "Overview of Methodologies for Building Ontologies." In IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm.

Fernández-López, M. and A. Gómez-Pérez. 2002. "Overviews and analysis of methodologies for building ontologies." The Knowledge Engineering Review 17(2):129-156.

Fernandez-Lopez, Mariano., Asuncion. Gomez-Perez and N. Juristo. 1997. "METHONTOLOGY: From Ontological Art Towards Ontological Engineering " In AAAI Symposium on Ontological Engineering. Stanford.

Fernández-López, M. F., A. Gomez-Perez, J. P. Sierra and A. P. Sierra. 1999. "Building a chemical ontology using Methontology and the Ontology Design Environment." Intelligent Systems and their Applications, IEEE 14(1):37-46.

Fischer, Ulli and Dragan Stokic. 2002. "Organisational Knowledge Management in Manufacturing Enterprises-Solutions and Open Issues." In The project PICK.

Frankovič, Baltazár and Ivana Budinská. 2006. "The Role of Ontology in Building of Knowledge Systems for Industrial Applications." In 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence. Herlany, Slovakia.

Gabel, Thomas, York Sure and Johanna Voelker. 2004. "KAON – Ontology Management Infrastructure." Institute AIFB, University of Karlsruhe.

Gallagher, C.C. and W.A. Knight. 1986. "Group technology production methods in manufacture." Ellis Horwood, Chichester UK.

Gallaher, Michael P, Alan C O'Connor, John L Dettbarn, Jr. and Linda T Gilday. 2009. "Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry."

- Gaithersburg, Maryland: U.S. Department of Commerce Technology Administration, National Institute of Standards and Technology (NIST).
- Gangemi, Aldo. 2005. "Ontology Design Patterns for Semantic Web Content." In *The Semantic Web* – ISWC 2005.
- Gangemi, Aldo and Stefano Borgo. 2004. "Core Ontologies in Ontology Engineering 2004 (Un)Successful cases and best practices for ontology engineering: reusing well-founded ontologies for domain content specification." In *Proceedings of the EKAW*04 Workshop on Core Ontologies in Ontology Engineering*. Northamptonshire (UK).
- Genesereth, Micheal R. and Nils. J Nilsson. 1987. *Logical Foundation of Artificial Intelligence*: Morgan Kaufmann Publishers Inc.
- Goffin, Keith and Ursula Koners. 2011. "Tacit Knowledge, Lessons Learnt, and New Product Development." *Journal of Product Innovation Management* 28(2):300-318.
- Gomez-Perez, A. 1998. "Knowledge sharing and reuse." In *Handbook of Expert Systems*, CRC accessed online on 17/06/2011 at <http://www.suchitav.com/artificial-intelligence-3/>. J. Liebowitz Edition. NewYork.
- Gomez-Perez, A. 2001. "Evaluation of ontologies." *International Journal of Intelligent Systems* 16(3):10:11–10:36.
- Gómez-Pérez, Asunción. 1996. "Towards a framework to verify knowledge sharing technology." *Expert Systems with Applications* 11(4):519-529.
- Gómez-Pérez, A., M. Fernández-López and O. Corcho. 2004. "Ontological engineering: with examples from the areas of knowledge management." In *e-commerce and the semantic web*. London, UK: Springer-Verlag.
- Gómez-Pérez, Asunción and Ma Rojas-Amaya. 1999. "Ontological Reengineering for Reuse." In *Knowledge Acquisition, Modeling and Management*, eds. Dieter Fensel and Rudi Studer: Springer.
- Gonzalez-Perez, Cesar and Brian Henderson-Sellers. 2006. "A powertype-based metamodeling framework." *Software and Systems Modeling* 5(1):72-90.
- Grangel, Reyes, A C. Métral, Anne-Françoise Cutting-Decelle, Jean Pierre Bourey and R. I. M. Young. 2007. "Ontology Based Communications Through Model Driven Tools: Feasibility of the MDA Approach in Urban Engineering Projects." In *Ontologies for Urban Development*, eds. Jacques Teller, John R. Lee and Catherine Roussey: I Springer.
- Gruber, Thomas, R. 1995. "Toward principles for the design of ontologies used for knowledge sharing." Academic Press, Inc.
- Gruninger, M. and M. Fox. 1995. "Methodology for the Design and Evaluation of Ontologies." In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*, April 13, 1995.

- Gruninger, M. and J. Lee. 2002. "Introduction: Ontology Applications and Design." *Communications of the ACM* 45(2).
- Guarino, Nicola. 1997. "Understanding, building and using ontologies." Academic Press, Inc.
- Guarino, N. and P. Giaretta. 1995. "Ontologies and Knowledge Bases: Towards a Terminological Clarification." *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*:25-32.
- Guarino, Nicola and Christopher Welty. 2000. "Ontological analysis of taxonomic relationships." In *Proceedings of the 19th international conference on Conceptual modeling*. Salt Lake City, Utah, USA: Springer-Verlag.
- Guarino, Nicola and Christopher A. Welty. 2000. "A Formal Ontology of Properties." In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*: Springer-Verlag.
- Gunendran, AG. 2004. "An Information and Knowledge Framework to Support Multiple Viewpoints in the Design for Manufacture of Injection Moulded Products, PhD Research Thesis, ." Loughborough University.
- Gunendran, A. G. and R. I. M. Young. 2010. "Methods for the capture of manufacture best practice in product lifecycle management." *International Journal of Production Research* 48(20):5885-5904.
- Gunendran, George and Bob Young. 2008. "Methods for the Capture of Manufacture Best Practice in Product Lifecycle Management." In *International Conference on Product Lifecycle Management 2008*: anderscience publishers.
- Haendel, M.A., F. Neuhaus, D. Osumi-Sutherland, P.M. Mabee, J.L.V. Mejino, C.J. Mungall and B. Smith. 2008. "CARO – the common anatomy reference ontology." *Computational Biology* 6(4):327-349.
- Hakimpour, Farshad. 2003. "Using Ontologies to Resolve Semantic Heterogeneity for Integrating Spatial Database Schemata." Zurich: University of Zurich.
- Han, J.H. and A.A.G. Requicha. 1995. "Integration of Feature Based Design and Feature Recognition." In *Proceedings of ASME 15th International Conference in Engineering Conference*. Boston, MA, USA.
- Hansen, Morten T. 2002. "Knowledge Networks: Explaining Effective Knowledge Sharing in Multiunit Companies." *Organization Science* 13(3):232-248.
- Harding, J. A. 1996. "A knowledge representation model to support concurrent engineering teamwork." In *Mechanical & Manufacturing Engineering*. Loughborough: Loughborough University.
- Harding, J. A., K. Popplewell and D. Cook. 2003. "Manufacturing system engineering

- moderator: an aid for multidiscipline project teams." 41(9):1973 - 1986.
- Harding, J. A. and B. Yu. 1999. "Information-centred enterprise design supported by a factory data model and data warehousing." Elsevier Science Publishers B. V.
- Heijst, G. van, A. Th. Schreiber and B. J. Wielinga. 1997. "Using explicit ontologies in KBS development." Academic Press, Inc.
- Henderson-Sellers, B. 2011. "Bridging metamodels and ontologies in software engineering." *Journal of Systems and Software* 84(2):301-313.
- Henderson-Sellers, Brian and Cesar Gonzalez-Perez. 2005. "Connecting Powertypes and Stereotypes." *The Journal of Object Technology* 4(7):83-96.
- Henderson-Sellers, Brian and Igor Hawryszkiewicz. 2008. "Comparing collaborative and process semantics for cooperative information systems." *International Journal of Cooperative Information Systems* 17(2):155-176.
- Hendriks, Paul. 1999. "Why share knowledge? The influence of ICT on the motivation for knowledge sharing." *Knowledge and Process Management* 6(2):91-100.
- Hoffmann, Patrick, Shaw C. Feng, Gaurav Ameta, Parisa Ghodous, Lihong Qiao, Richard Curran, Shuo-Yan Chou and Amy Trappey. 2008. "Towards a Multi-View Semantic Model for Product Feature Description Collaborative Product and Service Life Cycle Management for a Sustainable World." Springer London.
- Hoque, A. S. M. and T. Szecsi. 2008. "Designing using manufacturing feature library." *Journal of Materials Processing Technology* 201(1-3):204-208.
- Houtzeel, A. 1975. "MICLASS-A classification system based on group technology." SME Technical Paper:MS 75-79.
- Huang, T. T., R. A. Stewart and L. Chen. 2009. "Knowledge sharing leveraging new product development activities to derive enhanced business performance: Mixed method study." In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*.
- IDEAS. 2003. "Project Deliverables (WP1-WP7), Public Reports, www.ideas-roadmap.net".
- IDEF0. 1993. "Integration DEFINITION for Function modelling (IDEF0)." In National Institute of Standards and Technology (NIST), ed. Federal Information Processing Standards Publications (FIPS PUBS): Air Force Wright Aeronautical Laboratories Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part II, Volume IV.
- IEEE-Std-Computer-Dictionary. 1991. "IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries." In IEEE Std 610.
- Ilkka, Tuomi. 1999. "Data is more than knowledge: implications of the reversed knowledge

hierarchy for knowledge management and organizational memory." J. Manage. Inf. Syst. 16(3):103-117.

IMKS. 2011. "Interoperable Manufacturing Knowledge System (IMKS), ." <http://www.lboro.ac.uk/departments/mm/research/product-realisation/imks> (09-12-2011).

IODE, Highleet. 2010. "Ontology Libraray Refernce." In Highfleet's IODE/XKS Documentation. Baltimore.

ISO-10303-1. 1994. "Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles." Inetrnqational Organisation of Standarads.

ISO-10303-48. 1992. "STEP Product data representation and exchange, Integrated generic resource: Form Feature." Inetrnqational Organisation of Standarads, Sub-Committe 4, Part 47, NIST.

ISO-10303-49. 1998. "Industrial automation systems and integration -- Product data representation and exchange -- Part 49: Integrated generic resources: Process structure and properties." Inetrnqational Organisation of Standarads.

ISO-10303-108. 2005. "Industrial automation systems and integration -- Product data representation and exchange -- Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models."

ISO-10303-224. 2006 "Industrial automation systems and integration -- Product data representation and exchange -- Part 224: Application protocol: Mechanical product definition for process planning using machining features." Inetrnqational Organisation of Standarads.

ISO-10303-239. 2006. "Industrial automation systems and integration -- Product data representation and exchange -- Part 239: Application protocol: Product Life-cycle support." Inetrnqational Organisation of Standarads.

ISO-13399. 2006. "Cutting tool data representation and exchange. ." Inetrnqational Organisation of Standarads.

ISO-13584. 2001. "Industrial automation systems and integration – Parts library. ." Inetrnqational Organisation of Standarads.

ISO-15531-1. 2004. "Industrial automation systems and integration – Industrial manufacturing management data – Part 1: General overview. ."

ISO-15531-43. 2006. "Industrial automation systems and integration -- Industrial manufacturing management data -Part 43: Manufacturing flow management data: Data model for flow monitoring and manufacturing data exchange." Inetrnqational Organisation of Standarads.

ISO-16100-1. 2009. "Industrial automation systems and integration -- Manufacturing

software capability profiling for interoperability -- Part 1: Framework." International Organisation of Standards.

ISO-19439. 2006. "Industrial automation systems – Framework for enterprise modelling." International Organisation of Standards.

ISO/CEN-11354. 2008. "Requirements for establishing manufacturing enterprise process interoperability: Part 1: Framework for Enterprise Interoperability."

ISO/IEC-24707. 2007. "International Standard, First edition 2007-10-01 Information technology — Common Logic (CL): a framework for a family of logic based languages."

ISO/IEC-62264. 2002. "Enterprise-control system integration, Part 1. Models and terminology, Part 2: Model object attributes."

ISO/IEC-TR-10000-1. 1998. "Information technology -- Framework and taxonomy of International Standardized Profiles -- Part 3: Principles and Taxonomy for Open System Environment Profiles."

ISO/IEC-TR-10000-3. 1998. "Information technology -- Framework and taxonomy of International Standardized Profiles -- Part 3: Principles and Taxonomy for Open System Environment Profiles."

ISO/TS-10303-1017. 2004. "Industrial automation systems and integration -- Product data representation and exchange --Part 1017: Application module: Product identification".

ISO/TS-10303-1018. 2004. "Industrial automation systems and integration -- Product data representation and exchange -- Part 1018: Application module: Product Version."

ISO/TS-10303-1022. 2004. "Industrial automation systems and integration -- Product data representation and exchange -- Part 1022: Application module: Part and Version Identification."

ISO/TS-10303-1164. 2004. "Industrial automation systems and integration -- Product data representation and exchange -- Part 1164: Application module: Product as Individual."

ISO/TS-10303. 2004. "STEP modules related to Product Data Management. Industrial automation systems and integration—Product data representation and exchange."

ISO/TS-15926-4. 2007. "Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 4: Initial reference data."

ISO-IS-18629-1. 2004. "Industrial Automation System and Integration—Process Specification Language: Part 1. Overview and Basic Principles."

ISO-IS-18629-11. 2003. "Industrial Automation System and Integration—Process Specification Language: Part 11. PSL Core."

ISO-IS-18629-12. 2003. "Industrial Automation System and Integration—Process

Specification Language: Part 12. Outer Core."

Jagenberg, Jan Tim, Erik A. Gilsdorf, Reiner Anderl and Thomas Bornkessel. 2009. "KNOWLEDGE DRIVEN DESIGN FEATURES FOR THE PRODUCT LIFE CYCLE OF ENGINE PARTS." In Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009 San Diego, California, USA.

Jan de Kraker, K. 1998. "Feature Conversion for Concurrent Engineering." In Computer Science. Delft University of Technology: Delft, The Netherlands.

Johanson, U. 2008. "Industry, Trade and Services, eurostat-statistics in focus, 37/2008."

Julio, C. Arpirez, Corcho Oscar, Fern Mariano, L. ndez, pez, Asunci, G. n, P. mez and rez. 2001. "WebODE: a scalable workbench for ontological engineering." In Proceedings of the 1st international conference on Knowledge capture. Victoria, British Columbia, Canada: ACM.

Jung, J.Y. and R.S.. Ahluwalia. 1991. "A coding and classification system for formed parts." Journal of Manufacturing Systems 10(3):223-232.

Keqin, Wang and Tong Shurong. 2008. "An Ontology of Manufacturing Knowledge for Design Decision Support." In Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on.

Kitamura, Yoshinobu. 2006. "Roles Of Ontologies Of Engineering Artifactsfor Design Knowledge Modeling." In Proc. of the 5th International Seminar and Workshop Engineering Design in Integrated Product Development (EDIPrOD 2006). Gronów, Poland.

Koestler, A. 1967. "The Ghost in the Machine." Hutchinson, London.

Kosanke, K., F. Vernadat and M. Zelm. 1999. "CIMOSA: enterprise engineering and integration." Computers in Industry 40(2-3):83-97.

Kuffner, Tom A. and David G. Ullman. 1991. "The information requests of mechanical design engineers." Design Studies 12(1):42-50.

Kwon, Ohbyung, Kyoung-Yun Kim and Kun Chang Lee. 2007. "MM-DSS: Integrating multimedia and decision-making knowledge in decision support systems." Expert Systems with Applications 32(2):441-457.

Lagos, N. and R. M. Setchi. 2007. "A Manufacturing Ontology for E-Learning." In Proceedings of the 3rd Intelligent Production Machines and Systems Conference. Cardiff, UK: PROMS Network of Excellence.

Lee, Byungwoo and Kazuhiro Saitou. 2002. "Design of part family robust-to-production plan variations based on quantitative manufacturability evaluation." Research in Engineering Design 13 199–212.

- Lee, Jaehyun, Hyowon Suh and Soon Hung Han. 2005. "Ontology-based Knowledge Framework for Product Development." *Computer Aided Design & Applications* 2(5):635-643.
- Lee, Jae-Hyun and Hyo-Won Suh. 2008. "Ontology-based Multi-layered Knowledge Framework for Product Lifecycle Management." *Concurrent Engineering* 16(4):301-311.
- Lee, Jae-Nam. 2001. "The impact of knowledge sharing, organizational capability and partnership quality on IS outsourcing success." *Information & Management* 38(5):323-335.
- Leimagnan, Severin, Ali Siadat, Jean-Yves Dantan and Anatoli Semenenko. 2006. "MASON: A proposal for an ontology for manufacturing domain." In *Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*: IEEEExplore.
- Lemrabet, Youness, Michel Bigand, David Clin, Nordine Benkeltoum and Jean-Pierre Bourey. 2010. "Model driven interoperability in practice: preliminary evidences and issues from an industrial project." In *Proceedings of the First International Workshop on Model-Driven Interoperability*. Oslo, Norway: ACM.
- Lenat, D., B., and R. Guha, V.,. 1990. "Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project." *Artificial Intelligence*.
- Lin, H., J. Harding and P. Teoh. 2005. "An inter-enterprise semantic web system to support information autonomy and conflict moderation." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219(12):903-911.
- Lin, Hsiu-Fen 2007. "Knowledge sharing and firm innovation capability: an empirical study." *International Journal of Manpower* 28 (3/4):315 - 332.
- Lin, H. K. and J. A. Harding. 2007. "A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration." *Computers in Industry* 58(5):428-437.
- Lin, H. K., J. A. Harding and M. Shahbaz. 2004. "Manufacturing system engineering ontology for semantic interoperability across extended project teams." Taylor & Francis.
- Liping, Zheng, Li Guangyao, Liang Yongquan and Sha Jing. 2007. "Design of ontology mapping framework and improvement of similarity computation." *Journal of Systems Engineering and Electronics* 18(3):641-645.
- Lorre, Jean-Pierre, Yiannis Verginadis, Nikos Papageorgiou and Nicolas Salatge. 2010. "Ad-hoc Execution of Collaboration Patterns using Dynamic Orchestration." In *Enterprise Interoperability IV*, eds. Keith Popplewell, Jenny Harding, Raul Poler and Ricardo Chalmeta. Coventry, UK: Springer London.
- Love, D.M. 1985. "Flexibility in the design and application of component coding and classification software." In *National Conference on Production Research*. Nottingham, UK.
- Maedche, A. 2003. *Ontology Learning for the Semantic Web*. Norwell: Kluwer Academic

Publishers.

Maedche, A. and S. Staab. 2001. "Ontology learning for the Semantic Web." *Intelligent Systems*, IEEE 16(2):72-79.

Manufacture, A Vision for 2020. 2004. "Manufacture, A Vision for 2020, Assuring the Future of Manufacturing in Europe." European Commission Report of the High-level Group.

Martin, P and A. D'Acunto. 2003. "Design of a production system: an application of integration product-process." *International Journal of Computer Integrated Manufacturing* 16:509-516.

Masolo, Claudio, Stefano Borgo, Aldo Gangemi, Nicola Guarino and Alessandro Oltramari. 2003. "Ontology Librarian." In *WonderWeb Project*. Italy: Laboratory For Applied Ontology - ISTC-CNR.

Mesmer-Magnus, Jessica R. and Leslie A. DeChurch. 2009. "Information Sharing and Team Performance: A Meta-Analysis." *Journal of Applied Psychology* 94(2):535-546.

Miller, G.A. 1995. "WordNet: A Lexical Database for English." *Communications of the ACM (CACM)* 38(11):39-41.

Ming, X., J. Yan, W. Lu, D. Ma and B. Song. 2007. "Mass production of tooling product families via modular feature-based design to manufacturing collaboration in PLM." *Journal of Intelligent Manufacturing* 18(1):185-195.

MISSION-Consortium. 2001. "Final Report from MISSION Project—Deliverable D24.K, eds " K. Popplewell, J. A. Harding and M. Rabe [www.ims-mission.de].

Mizoguchi, R. 2003. "Tutorial on ontological engineering - Part1." *New Generation Computing* 21(4):365-384.

Mizoguchi, R. 2004. "Tutorial on ontological engineering - Part 2." *New Generation Computing* 22(1):61-96.

Mizoguchi, R. 2004. "Tutorial on ontological engineering - Part 3." *New Generation Computing* 22(2):193-220.

Moalla, N., Y. Ouzrout, G. Neubert and A. Bouras. 2008. "Model-driven interoperability to enhance product data quality." In *Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) in Conjunction with the CAISE'08 Conference*. Montpellier, France.

Molina, A and Bell R. 1999. "A Manufacturing Model representation of a flexible manufacturing facility." *Journal of Engineering Manufacture: Proceedings of the Institution of Mechanical Engineers* 213, Part B:225-246.

Niles, I., Pease, A. 2001. "Towards a Standard Upper Ontology." In *Teknowledge Corporation*. Palo Alto, CA

- Nof, S. Y., G. Morel, L. Monostori, A. Molina and F. Filip. 2006. "From plant and logistics control to multi-enterprise collaboration." *Annual Reviews in Control* 30(1):55-68.
- Nonaka, Ikujiro and Hirotaka Takeuchi. 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford: Oxford University Press.
- Noy, Natalya F. . 2004. "Tools for Mapping & Merging Ontologies." In *Handbook on Ontologies*, eds. Steffen Staab and Rudi Studer: Birkhäuser.
- Noy, Natalya F. and Deborah L. McGuinness. 2000. "Ontology Development 101: A Guide to Creating Your First Ontology." ed. Deborah L. McGuinness.
- Oberle, Daniel. 2004. "Semantic management of middleware." In *Proceedings of the 1st international doctoral symposium on Middleware*. Toronto, Ontario, Canada: ACM.
- Oberle, Daniel, Anupriya Ankolekar, Pascal Hitzler, Philipp Cimiano, Michael Sintek, Malte Kiesel, Babak Mougouie, Stephan Baumann, Shankar Vembu, Massimo Romanelli, Paul Buitelaar, Ralf Engel, Daniel Sonntag, Norbert Reithinger, Berenike Loos, Hans-Peter Zorn, Vanessa Micelli, Robert Porzel, Christian Schmidt, Moritz Weiten, Felix Burkhardt and Jianshen Zhou. 2007. "DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO)." *Web Semantics: Science, Services and Agents on the World Wide Web* 5(3):156-174.
- OpenCyc. accessed 12-04-2011. "<http://www.opencyc.org/>."
- Opitz, H. 1970. "A classification system to describe workpieces, Part I and II." Pergamon Press.
- Oxford-Dictionary. 2012. "www.oxforddictionaries.com." Accessed 19-12-2010.
- Ozman, Müge. 2006. "Knowledge integration and network formation." *Technological Forecasting and Social Change* 73(9):1121-1143.
- Palmer, C., A.G. Gunendran, R.I.M. Young, N. Chungoora, Z. Usman, K. Case and J.A. Harding. 2012. "Exploiting UML as a Preliminary Design Tool for Common Logic-based Ontologies." *International Journal of Computer Integrated Manufacturing*.
- Panetto, Hervé and Arturo Molina. 2008. "Enterprise integration and interoperability in manufacturing systems: Trends and issues." *Computers in Industry* 59(7):641-646.
- Patil, L., D. Dutta and R. Sriram. 2005. "Ontology-based exchange of product data semantics." *Automation Science and Engineering, IEEE Transactions on* 2(3):213-225.
- Patil, Lalit, Debasish Dutta and Ram D. Sriram. 2005. "Ontology Formalization of Product Semantics for Product Lifecycle Management." In *Manufacturing Systems Integration Division*, National Institute of Standards and Technology (NIST). Gaithersburg.
- PERA. 1969. "Preliminary study of geometric-based system of component description." PERA production system report-207.

- Porter, Micheal, E., . 1990. *Competitive Advantage of Nations*. New York: Free Press.
- Pouchard, Line, Nenad Ivezic and Craig Schlenoff. 2000. "Ontology Engineering for Distributed Collaboration in Manufacturing." In AIS2000 conference.
- PRATT, M. J. 1993. "Applications of feature recognition in the product life-cycle." *International Journal of Computer Integrated Manufacturing Systems* 6(1-2):13-19.
- PSL. accessed 01/08/2011. "<http://www.mel.nist.gov/psl/download.html>."
- Rabe, M. 2000. "Future of simulation in production and logistics: facts and visions." In *The New Simulation in Production and Logistics*. 9th ASIM Dedicated Conference on Simulation in Production and Logistics, ed. Mertins and M. Rabe (eds). Berlin, Germany.
- Ray, S. and A. Jones. 2006. "Manufacturing interoperability." *Journal of Intelligent Manufacturing* 17(6):681-688.
- Ray, S.R. 2004. "Semantic approach to standards and interoperability. PowerPoint presentation. Available from: <http://ontolog.cim3.net/file/resource/presentation/> accessed 18th May 2011." National Institute of Standards and Technology (NIST).
- Ray, S.R. and A.T. Jones. 2003. "'Manufacturing interoperability", *Concurrent Engineering: Enhanced Interoperable System*." In 10 th ISPE international Conference. Madeira Island - Portugal.
- Reid, Frank. 2003. "Creating a knowledge-sharing culture among diverse business units." *Employment Relations Today* 30(3):43-49.
- Roche, Christophe. 2000. "Corporate ontologies and concurrent engineering." *Journal of Materials Processing Technology* 107(1-3):187-193.
- Ruggaber, R. 2006. "ATHENA – Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications ." In *Interoperability of enterprise software and applications (IESA)*, ed. D. Konstantas, Bourrières, J.-P., Léonard, M., Boudjilida, N. London, UK: Springer-Verlag.
- Salomons, O. W., F. J. A. M. van Houten and H. J. J. Kals. 1993. "Review of research in feature-based design." *Journal of Manufacturing Systems* 12(2):113-132.
- Samer, Abdul-Ghafour, Ghodous Parisa, Shariat Behzad and Perna Eliane. 2007. "A Common Design-Features Ontology for Product Data Semantics Interoperability." In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence: IEEE Computer Society*.
- Sánchez, Diana Marcela , José María Cavero and Esperanza Marcos Martínez. 2007. "The road towards ontologies " In *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, ed. Rajiv Kishore and Ram Ramesh Raj Sharman:

Springer.

Sandvik. 2011. "Application guide: Heat Resistant super alloys."

Sarder, M.D.B. 2006. "The development of a design ontology for products and Processes." The University Of Texas At Arlington.

Schlenoff, C. , M. Gruninger, F. Tissot, J. Valois, J. Lubell and J. Lee. 2000. "ISO-18629 The Process Specification Language (PSL): Overview and Version 1.0 Specification," NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD., ."

Schreiber, A , B Wielinga and W. Jansweijer. 1995. "The KACTUS view on the _O_ word. Technical Report, ESPRITProject 8145 KACTUS." In University of Amsterdam. The Netherlands.

Schreiber, A. Th., Hans Akkermans, Anjo Anjewierden, Robber de Hoog, Nigel SHadbolt, Walter Van de Yslde and Bob Wielinga. 2000. Knowledge Engineering and Management: the commonKASD methodology.

Schreiber, G., B. Wielinga, R. de Hoog, H. Akkermans and W. Van de Velde. 1994. "CommonKADS: a comprehensive methodology for KBS development." IEEE Expert 9(6):28-37.

Semere, Daniel, T, Saqib Dislshad and Bengt Lindberg. 2007. "Machining ontology and knowledge modelling." In Swedish Production Symposium. Sweden.

Siemens-website. 2012.
["http://www.plm.automation.siemens.com/en_us/products/teamcenter/."](http://www.plm.automation.siemens.com/en_us/products/teamcenter/)

Simpson, J. A., R. J. Hocken and J. S. Albus. 1982. "The automated manufacturing research facility of the national bureau of standards." Journal of Manufacturing Systems 1(1):17-32.

Spek, Rob van der and André Spijkervet. 2005. Dealing Intelligently With Knowledge.

Staab, S., R. Studer, H. P. Schnurr and Y. Sure. 2001. "Knowledge processes and ontologies." Intelligent Systems, IEEE 16(1):26-34.

Staub, Peter, Hans Rudolf Gnägi and Andreas Morf. 2008. "Semantic Interoperability through the Definition of Conceptual Model Transformations." Transactions in GIS 12(2):193-207.

Steels, Luc, Guus Schreiber, Walter Van de Velde, Guus Schreiber, Bob Wielinga, Hans Akkermans, Walter Van de Velde and Anjo Anjewierden. 1994. "CML: The commonKADS conceptual modelling language." In A Future for Knowledge Acquisition: Springer Berlin / Heidelberg.

SUMO. accessed 12-04-2011. "Suggested Upper Merged Ontology "
[http://www.ontologyportal.org/.](http://www.ontologyportal.org/)

- Sura, Walter 2009 "Specialisations within EU manufacturing." In Industry, trade and services Statistics in focus
- Sureephong, P., N. Chakpitak, Y. Ouzrout and A. Bouras. 2008. "An Ontology-based Knowledge Management System for Industry Clusters." In International Conference on Advanced Design and Manufacture (ICADAM 2008). Sanya: China.
- Sveiby, Karl, Erick 1997. The New Organizational Wealth: Managing and Measuring Knowledge-Based Assets San Francisco: Berrett-Koehler Inc.
- Swartout, B., P. Ramesh, K. Knight and T. Russ. 1997. "Toward Distributed Use of Large-Scale Ontologies." AAAI Symposium on Ontological Engineering.
- Tan, Philip, Stuart E. Madnick and Kian-Lee Tan. 2004. "Context Mediation in the Semantic Web: Handling OWL Ontology and Data Disparity through Context Interchange." SSRN eLibrary.
- Terzi, Sergio , Hervé Panetto, Gérard Morel and Marco Garetti. 2007. "A holonic metamodel for product traceability in PLM " International Journal of product lifecycle management.
- Todd, Robert. H. 1994. "Fundamental principles of manufacturing processes." In Fundamental principles of manufacturing processes., ed. Industrial Press Inc. 1 Edition. New York: Industrial Press.
- Trade-policy-review-report. 2009. "Trade policy review report by European communities, World Trade Organization-Trade Policy Review ".
- True, M. and C. Izzi. 2002. "Collaborative Product Commerce: Creating Value Across the Enterprise." Accenture.
- Turban, E. and J. Aronson. 2005. Decision Support Systems and Intelligent Systems. 7 Edition: Prentice Hall, Inc.
- Tursi, A., H. Panetto, G. Morel and M. Dassisti. 2009. "Ontological approach for products-centric information system interoperability in networked manufacturing enterprises." Annual Reviews in Control 33(2):238-245.
- Uschold, Michael. 1996. "Building Ontologies: Towards a Unified Methodology." In Proceedings of Expert Systems '96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, . Cambridge UK.
- Uschold, Mike and Michael Gruninger. 1996. "Ontologies: principles, methods, and applications." Knowledge Engineering Review 11(2):93-155.
- Uschold, Michael and Michael Gruninger. 2004. "Ontologies and Semantics for Seamless Connectivity."
- Uschold, Michael and R Jasper. 1999. "A Framework for Understanding and Classifying

Ontology Applications." In IJCAI99 Workshop on Ontologies and Problem-Solving Methods. Stockholm.

Uschold, Michael and M King. 1995. "Towards a Methodology for Building Ontologies." In IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal.

Uschold, Mike, Martin King, Stuart Moralee and Yannis Zorgios. 1998. "The Enterprise Ontology." Knowledge Engineering Review 13(1):31-89.

Usman, Zahid, Muhammad Imran, Amjad Hussain, N.A. Anjum and Muahammad Kaiser Saleem. 2011. "A Heavyweight Ontological Approach To Support Interoperability Across Organizations." In International Conference on Advanced Modelling and Simulation. Rawalpindi, Pakistan.

Usman, Z. , Young R.I.M., N. Chungoora, C. Palmer, K. Case and J.A. Harding. 2011. "A Manufacturing Core Concepts Ontology for Product Life Cycle Interoperability " In International Ifip Working conference on Enterprise Interoperability. Stockholm, Sweden: Springer.

Usman, Zahid, Robert I.M. Young, Keith Case and Jenny A. Harding. 2010. "A Manufacturing Foundation Ontology for Product Life Cycle Interoperability." In Interoperability for Enterprise Software and Applications. Coventry, UK: Springer.

Usman, Zahid, Robert Ian Marr Young, Nitishal Chungoora, Claire Palmer, Keith Case and Jenny Harding. 2011. "A Manufacturing Core Concepts Ontology for Product Lifecycle Interoperability." In Enterprise Interoperability, eds. Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw and Clemens Szyperski. Stockholm, Sweden: Springer Berlin Heidelberg.

Vichare, Parag, Aydin Nassehi, Sanjeev Kumar and Stephen T. Newman. 2009. "A Unified Manufacturing Resource Model for representing CNC machining systems." Robotics and Computer-Integrated Manufacturing 25(6):999-1007.

Wang, Ming-Tzong, Marcella A. Chamberlain, Ajay Joneja and Tien-Chien Chang. 1993. "Manufacturing feature extraction and machined volume decomposition in a computer-integrated feature-based design and manufacturing planning environment." Computers in Industry 23(1-2):75-86.

Webster's online Dictionary, "Feature def1". 2011. "<<http://www.websters-online-dictionary.org>> (08-12-2011)."

Wiig, K.M. 1994. Knowledge Management Foundations: Thinking About Thinking - How People and Organizations Create, Represent, and Use Knowledge. Arlington, TX:: Schema Press.

Wordnet. 2010. "Princeton University "About WordNet." WordNet. Princeton University. <<http://wordnet.princeton.edu>> (08-12-2011)."

Yoshinobu, Kitamura. 2006. "Roles of ontologies of engineering artifacts for design knowledge modelling." In 5th International Seminar and Workshop Engineering Design in Integrated Product Development (EDIPrD-06). Gronów, Poland.

Young, R.I.M, Chungoora, N., Usman, Z., Anjum, N.A, Gunendran, G., Palmer, C., Harding, J.A, Case, K. and Cutting-Decelle, A.-F., . 2010. "An exploration of foundation ontologies and verification methods for manufacturing knowledge sharing." In Workshop on Interoperability for Enterprise Software and Applications (I-ESA). Coventry, UK: University of Coventry.

Young, R.I.M, Chungoora, N., Usman, Z., Anjum, N.A, Gunendran, G., Palmer, C., Harding, J.A, Case, K. and Cutting-Decelle, A.-F., . 2011. " Reference ontologies for manufacturing based ecosystems." In 3rd International IFIP Working Conference on Enterprise Interoperability (IWEI). Stockholm, Sweden.

Young, R.I.M, A. Gunendran, G. , A. Cutting-Decelle, F. and M. Gruninger. 2007. "Manufacturing knowledge sharing in PLM: a progression towards the use of heavy weight ontologies." International Journal of Production Research 45(7):1505–1519.

Yu, B. , J.A. Harding and K. Popplewell. 2000. "A reusable enterprise model " International Journal of Operations & Production Management 20(1):50 - 69.

Zhao, J., W.M. Cheung and R.I.M. Young. 1999. "A consistent manufacturing data model to support virtual enterprises." International Journal of Agile Management Systems 1(3):150-158.

Zhou, Jiehan and Rose Dieng-Kuntz. 2004. "Manufacturing Ontology Analysis and Design: Towards Excellent Manufacturing." IEEEExplore:39-45.

Zhu, Hongwei and Stuart E Madnick. 2007. "A Lightweight Ontology Approach to Scalable Interoperability." In Working paper- Massachusetts Institute of Technology (MIT), Sloan School of Management. Massachusetts, USA.

A1 An introduction to the Knowledge Frame Language

A1.1 Introduction

This document contains a description of the KFL ontology modeling language including properties, relations, rules and integrity constraints, functions, and constants.

This manual is a reference guide to the syntax and commonly used techniques for writing KFL files for use with the Integrated Ontology Development Environment (IODE). For a thorough introduction to the IODE, refer to the *Highfleet Tools User Manual*.

A1.2 What is KFL?

Knowledge Frame Language (KFL) is a convenience layer of syntactic sugar that sits on top of a base layer of logical syntax called Extended Common Logic Interchange Format (ECLIF). For convenience, we call the whole language (ECLIF + the syntactic sugar) “KFL”. A parenthesis-heavy syntax like ECLIF is confusing for some users. So, KFL is designed according to a “90% rule” – most of what an ontologist needs to do can be done without raw ECLIF. Most of KFL takes the form of *directives*. Directives are expressed as a colon at the start of a line, followed by a keyword and some arguments. Some arguments (like labels) are simply strings, but most are elements of the ontology already present on the server. This means that the tools used to create the ontology are themselves part of the ontology. The KFL you create builds on Upper Level Ontology (ULO) content preloaded on each new knowledge server. Writing effective KFL requires familiarity with the ULO, so this manual will introduce some of the ULO content alongside KFL. For more detail, refer to ULO Overview.

A1.3 Creating a KFL file

The IODE reads KFL files that describe the structure of your ontology. You can make them with any text editor. Here’s an example:

```
;;; File: my_ontology.kfl
;;; Date: 02 January 2006
:Name "my_ontology"
:Description "A simple ontology"
:Use MLO
:Prop Dog
:Inst Type
:sup Mammal
:name "Dog"
:rem "The class of carnivorous animals that includes wolves, jackals, foxes, coyote and dogs."
```

Any text preceded by the semicolon character “;” is a comment. Comments are never loaded into the ontology; they’re only there for anyone reading the KFL. The first directives above, :Name and :Description, are purely optional, and are also never loaded into the ontology.

(Earlier KFL specifications required these, and they are here for backward compatibility.) The :Use directive is mandatory. It declares the context that will prefix any entities defined later. You may have multiple :Use directives in a file. All terms defined in a KFL file are considered to be within the context named by the most recent

:Use directive. In the above example, the property name Dog will be in the MLO context.

A1.3.1 Contexts

The above example shows you how to use a preexisting context. You can also create custom contexts to stake out a new namespace or make modules that make a large ontology more manageable. The following KFL creates a new context Chemistry whose super-context (:supCtx) is MLO (short for Mid-Level Ontology, a predefined context). Chemistry is an instance of UserContext, a special type of context. (Any context you define should be an instance of UserContext.) This declaration also supplies an optional descriptive remark (:rem) and name (:name).

```
:Ctx Chemistry
:Inst UserContext
:supCtx MLO
:name "Chemistry Context"
:rem "basic chemistry"
```

Once you have defined a context you can subsequently use it with a :Use directive:

```
:Use Chemistry
:Rel molecularWeight
:Inst BinaryRel
:Sig MolecularSpecies MassQuality
:name "has molecular weight"
:rem "the mass of one molecule of a molecular species"
```

Contexts dictate how a symbol is referenced later. The full name of the relation introduced above is Chemistry.molecularWeight. Since the :Use Chemistry directive is in effect, symbols appearing without context are understood to have the Chemistry context by default.

You can form a context hierarchy by including the supCtx directive in your context definition.

```
:supCtx MLO
```

Symbols are only qualified by that context directly above them. For example, we refer to Chemistry.molecularWeight, not MLO.Chemistry.molecularWeight.

A1.3.2 Properties

Properties make up the taxonomic component of any KFL ontology. Because our ontologies are semantic (modelling meaning), rather than set-theoretical (modeling categorizations), we use the term *property* rather than *class* or *category*.

A1.3.2.1 Useful Property Kinds

Most of the time in practice, you'll use two kinds of properties defined in the Upper Level Ontology: Type and MaterialRole. Types are properties that "stick" to their instances permanently. An instance of a type can't cease to be such while it exists. Material roles, in contrast, can come and go. Below are examples of declaring a type and a material role:

```
:Prop Person
:Inst Type
:sup ConcreteObject
:name "person"
:rem "A human being, living or dead."
:Prop Waiter
:Inst MaterialRole
:sup Person
:name "waiter"
:rem "A person hired to wait tables at a restaurant."
```

The :Prop, :Inst, and :sup directives are required for all properties. The :name and :rem directives are optional, though strongly recommended. The :Prop directive introduces the property. The :Inst directive states what the property instantiates (usually Type or MaterialRole). Hierarchies of properties are defined via the super-property relation, referred to as sup in KFL. Property A

is a super-property of property B if every instance of property B is necessarily an instance of property A. The :sup directive has some special properties:

- Its argument must be an instance of Property.
- A property cannot be a sub-property of itself.
- A property may have more than one direct super-property. In KFL, this may appear as multiple :sup directives.
- Types cannot be subsumed by material roles.

A1.3.3 Relations

Relations are the glue that holds objects together. Properties by themselves do not convey much. Properties are given meaning by the relationships between their instances. For example, the sup relation previously discussed defines a hierarchy of properties. Unlike UML and other modeling paradigms, KFL does not have "attributes"; we describe entities by relating them to each other and to primitive data values. A relation declaration in KFL is as follows:

```
:Rel brotherOf
:Inst BinaryRel
:Sig Person MalePerson
:Args "sibling" "brother"
:name "is a brother of"
```


Like properties, relations have three required fields in a declaration:

:Rel

:Inst

:Sig

Just as the :Inst line in a property declaration gave the kind of property, the :Inst line in a relation declaration gives the kind of relation. The arity (number of arguments) of a relation must be specified. In this case, the new relation is an instance of the type BinaryRel, meaning that it will relate two properties and have two argument places. In addition to BinaryRel, four other properties classify relations by number of arguments:

UnaryRel (one place)

TernaryRel (three place)

QuaternaryRel (four place)

QuinaryRel (five place)

Arity is not the only characteristic you might use to select what to supply to an :Inst directive, but it is the most important, and convenient as well. Furthermore, you can supply more than one :Inst directive, if your relation fits more than one kind. If one kind is implied by another, you can omit the :Inst for the first one, as it is redundant information otherwise. The :Sig line must have a property for each argument position. It restricts the arguments of that relation to objects that are instances of those properties.

A1.3.4 MetaProperty

Properties whose instances may only be properties are known as *meta properties*. Meta-properties come in handy for a large number of modelling tasks. For example, you might declare Airplane to be a super-property of Boeing747 and Boeing747 to be an instance of AirplaneModel. In this case AirplaneModel would be a meta-property. You'd declare it as follows:

:Prop AirplaneModel

:Inst MetaProperty

:sup Type

:name "airplane model"

It is common for a meta-property M to relate to a (non-meta-) property P in a special way. In the above example, all instances of AirplaneModel ought to be sub-properties of Airplane. You can enforce this by adding the line: :metaPropFor Airplane to the above example. This means that any instance of AirplaneModel must either be the property Airplane or a sub-property of Airplane.

A1.3.5 Intensional and Extensional Relations

Normally, all relations in an ontology may have assertions both concluded from rules, and asserted directly. (Rules are explained in the next chapter.) For example, you might assert that Jerry and Beth are cousins by inserting (hasCousin jerry beth) to the database; meanwhile, the database might also conclude (hasCousin mary allen) from assertions that Mary's and Allen's parents are siblings, and a rule that concludes cousinhood from those facts.

In limited cases, you might wish to restrict the manner in which certain relations may acquire assertions. (Such restrictions could potentially be used to optimize queries involving these relations.) A relation populated only by explicit assertions is an `ExtensionalRel`. A relation populated only by facts

concluded from rules is an `IntensionalRel`.

```
:Rel canCommunicateWith
:Inst BinaryRel
:Inst IntensionalRel
:Sig CommsPlatform CommsPlatform
:Rel friendsWith
:Inst SymmetricBR
:Inst ExtensionalRel
:Sig Person Person
```

A1.3.6 Functions

Functions are used to produce additional entities from one or more parameters. For instance, we often refer to masses, volumes, and speeds with measures. We can refer to places by their latitude and longitude, and to some companies by their NASDAQ symbol. Entities like `ninePointTwoGrams` and `fortyDegreesNByTwelveDegreesE` would quickly clutter the model, as well as obscure the parameters. Functions semantically distinguish between a description and what is described, and permit parameters to be used in reasoning. Here's an example of a function definition for grams::

```
:Fun gram
:Inst UnaryFun
:Sig RealNumber -> MassQuality
```

This lets us later write expressions that denote masses; for example: (gram 3.25). In general, a function term is written as the function name, followed by its arguments in order, with parentheses enclosing the entire term. Here is a latitude/longitude function mapping two real numbers (presumably denoting degrees) to a region:

```
:Fun latlong
:Inst BinaryFun
:Sig RealNumber RealNumber -> Region
```

Like relations and properties, functions have three mandatory fields:

```
:Fun
```

:Inst
:Sig ->

Other than the primary directive, the biggest difference between a function declaration and a relation declaration is the arrow in the :Sig directive. Values on the left of the arrow indicate the arguments to the function; the value on the right of the arrow is the property instantiated by the entire function term. For example, the sole argument to the gram function above is required to be an instance of RealNumber, and the whole term, (gram 3.25), is an instance of MassQuality. The return property of a function is necessarily true of a term constructed using that function if and only if the arguments in the term are of the respective types defined in the signature of the function. As a consequence, if any of arguments of a function are NonLogical (IntegerNumber, for example), then the query asking for all instances of the return property will be unbound.

Like relations, functions may instantiate properties which classify them by arity:

- UnaryFun – one argument
- BinaryFun – two arguments
- TernaryFun – three arguments
- QuaternaryFun – four arguments
- QuinaryFun – five arguments

A1.4 Documentation

Certain relations are used primarily to provide documentation to the reader, rather than facilitate reasoning within the model. Two of them – name and rem – are the most commonly used, and have been mentioned previously. Here is a list of the relations you can use to document entities in KFL.

- **name** – a string representing an entity, as it is likely to appear in English text.
- **abbrevString** – like a name, but abbreviated. Not every entity requires an abbrevString.
- **copyright** – denotes the copyright holder of a concept.
- **lex** – an English rephrasing of a term built on a property, relation or function, incorporating the arguments supplied to it. An argument is denoted by a question mark followed by its numerical position.
- **rem** – short for “remark”; a comment summarizing the meaning of an entity in readable text. Rems may contain fairly large amounts of text. They can span multiple lines, and may contain double quotes if they are escaped with a backslash (“”).
- **exampleRem** – a phrase intended to denote an example of an entity.
- **limitationRem** – a remark intended to explain constraints on the meaning of an entity, or in other words, what an entity does *not* mean.
- **referenceRem** – a remark citing the external reference material from which an entity was derived. The following relation provides an example of each.

:Rel tangentialProperPartOf
:Inst BinaryRel
:Sig Region Region

```

:name "tangential proper part"
:lex "?1 is a tangential proper part of ?2"
:Args "part" "whole"
:abbrevString "tpp"
:copyright company-42
:rem "A spatial part that is not identical with the whole region, yet shares a boundary with the whole."
:exampleRem "Texas is a tangential proper part of the US"
:limitationRem "Tangential proper parthood is only defined for regions, so a person's skin cannot be said
to be a tangential proper part of his body."
:referenceRem "1996 Ooley and Tooley, 'Spatial Parts', Journal of Geospatial Ontology"

```

A1.5 More on Directives

KFL directives are divided into two types. Directives such as :Ctx, :Use, and :Rel are *primary directives*; they are standalone, requiring no directive immediately preceding them. They are typically preceded by a blank line for readability. (The KFL parser does not require one.) Other primary directives include :Prop for properties, :Fun for functions, and :Const for constants; these are explained later.

All other directives are *secondary*; their meaning is dependent upon the most recent primary directive. For example, the :Inst BinaryRel directive above specifies that molecularWeight is a binary relation, and relates to no other directives elsewhere in the KFL file. The documentation directives described above are not defined in KFL at all; they are actually based on the ULO relations RootCtx.name and RootCtx.rem, respectively. The following are equivalent:

```

:Rel exampleRelation
:Inst BinaryRel
:rem "An example relation."
:Rel exampleRelation
:Inst BinaryRel
(rem exampleRelation "An example relation.")

```

The :rem directive above simply allows the writer to avoid having to write exampleRelation more than once.

A1.6 Axiomatisation

Beyond relations and properties, ontologies gain a lot of value from axioms i.e. logical rules. Rules allow new information to be deduced from existing facts. Rules can also act as constraint that prevent inconsistent facts. Like special relation types, constraints not only improve data quality, but are also used by the system to speed up query response times. In fact, the special relation types just discussed are simply shorthand for logical constraints that often arise in the modeling process.

A1.6.1 Variables

Every nontrivial rule and constraint uses variables in its expression. Variables look like ECLIF symbols that begin with a question mark. They work the same way in rules and constraints as they do in ECLIF queries: each variable is treated as a blank in which to substitute a value from the assertions in

the ontology, and if the same variable appears twice within a given rule or constraint, each appearance must use the same value substitution.

A1.6.2 Logical Operators

There are six primary logical operators in ECLIF used to write logical rules and constraints.

A1.6.2.1 Implication

Implication is the basic building block for rules. The implication operator is an arrow made up of an equals sign followed by a greater-than sign. This operator always takes two arguments. The first argument is called the *antecedent*. The second argument is called the *conclusion*.

Antecedents and conclusions are *clauses*; they look like ECLIF queries. (Consult the ECLIF Syntax Reference for more information about queries and variables.) If the antecedent is true for a given variable assignment, or *binding*, then the conclusion is also true for that binding. For example:

; A father of someone is also that person's parent.

```
(=> (fatherOf ?x ?y)
```

```
(parentOf ?x ?y))
```

With the rule above, for every fatherOf fact the server finds with some value assignment to ?x and ?y, a parentOf fact is also true for that same ?x and ?y. If you queried the server for the parent of Allen, this rule would give you the father of Allen. Implications can also be written in reverse. The reverse implication operator is <=. It takes the conclusion first and the antecedent second. The meaning does not change; just as “if P is true, then Q is true” means the same as “Q is true if P is true”.

```
(<= (parentOf ?x ?y)
```

```
(fatherOf ?x ?y))
```

(Both of the above examples could of course be replaced by a :supRel directive in KFL, making parentOf the superrelation of fatherOf. This is what supRel means. Implication rules that are this simple will often have such replacements.)

A1.6.2.2 Conjunction

A conjunction combines two or more clauses and is true when all clauses are true. The conjunction operator is and. A

conjunction is itself a clause.

```
; A parent of a parent is a grandparent.
(=> (and (parentOf ?x ?y)
  (parentOf ?y ?z))
  (grandparentOf ?x ?z))
; Children of sibling parents are cousins.
(=> (and (parentOf ?child1 ?parent1)
  (parentOf ?child2 ?parent2)
  (hasSibling ?parent1 ?parent2))
  (hasCousin ?child1 ?child2))
```

A1.6.2.3 Disjunction

A disjunction takes two or more clauses, and is true whenever at least one of those clauses is true.

```
(or (connectedTo ?x ?y)
  (disconnectedFrom ?x ?y))
; Everything is connected to or disconnected from ; everything else.
```

A1.6.2.4 Negation

The negation operator not takes only one clause, and is true whenever that clause is false.

```
; Steve is not Mary's father.
(not (hasFather Mary Steve))
```

A1.6.2.5 Universal Quantification

The universal quantification operator forall takes a variable specification and a clause. It is true whenever every binding for the variables in the specification makes the clause true.

```
; Everyone knows about a completely popular person.
(=> (forall (?y) (=> (Person ?y)
  (knowsAbout ?y ?x)))
  (CompletelyPopular ?x))
```

A1.6.2.6 Existential Quantification

The existential quantification operator exists takes a variable specification and a clause. It is true whenever at least one binding can be found for the variables in the specification which makes the clause true.

```
; Every US state has two (distinct) senators.
(=> exists (?y ?z) (and (senatorOf ?y ?x)
  (senatorOf ?z ?x)
  (/= ?y ?z)))
```

(US_State ?x))

A1.6.2.7 Integrity Constraints

Integrity constraints (ICs) look like rules, except that they are followed by an :IC directive. This directive indicates how strong the constraint should be, and what error message should be displayed when the constraint is violated.

IC strength is one of four values:

- *weak* – A violation should indicate an irregularity, but not necessarily a problem.
- *soft* – A violation should not prevent a transaction commit. This is stronger than a weak constraint.
- *hard* – A violation should rollback a transaction.
- *adamant* – A violation indicates assertions that could harm the integrity of the logic engine.

```
(=> (and (hasParent ?x ?y)
         (hasParent ?x ?z)
         (/= ?y ?z))
      (knowsWell ?y ?z))
```

:IC soft "The parents of a child should know each other well, but ?y and ?z do not."

```
(=> (and (disconnectedFrom ?x ?y)
         (hasPart ?x ?a)
         (hasPart ?y ?b))
      (disconnectedFrom ?a ?b))
```

:IC hard "Parts of disconnected regions should be disconnected, but ?a and ?b are not."

The integrity constraint condition expresses what should be true about the data. For example, the first IC above requires that if any ?x in the model has two parents recorded, ?y and ?z, who aren't equal, then it should be possible to prove (knowsWell ?y ?z), whether by having that fact explicitly stated, or derived from other facts and rules. (As far as ICs are concerned, there is no difference between extensional and intensional facts.) If, during an IC check, a counterexample to this constraint is found, then the IC fires and displays an error message.

ICs become active as soon as the KFL file containing them is completely loaded. They are checked against the content of the model up to that point.

A2 Formalisation of MCCO

This appendix presents the formalisation of Manufacturing Core Concepts Ontology

A2.1 Contexts for the MCCO and Application Specific Ontologies

```

;===== ;Context for Manufacturing Core Concepts Ontology =====
:Ctx MCCO
:Inst UserContext
:supCtx MLO
:name "MCCO"
:rem "MCCO denotes the context for building Manufacturing core concepts ontology"

;=====Context for Application Specific Ontologies =====
;=====Cpntext for Aero Engine Disc Production Ontology =====
:Ctx Production
:Inst UserContext
:supCtx MLO
:name "Production Context"
:rem "A context for concepts and instances to be defined exclusively from a Production viewpoint."

;=====;Context for Aero Engine Disc Design Ontology =====
:Ctx Design
:Inst UserContext
:supCtx MLO
:name "Design Context"
:rem "A context for concepts and instances to be defined exclusively from a design viewpoint."

;===== Relation to refer to the ICs pertaining to a specific context =====
:Use MLO
:Rel hasCtx
:Inst BinaryRel
:Sig OWLSentence UserContext
:rem "This relation should be used in situations where an integrity constraint or inference rule needs to be
referenced within a specific context."

```

A2.2 Generic Level Formalisation

This section details the generic level core concepts, relations, functions and axioms. Note that the generic level core concepts belonging to the ManufacturingMethod category are mentioned only in the section describing the formalisation of ManufacturingMethod. This is done to provide some ease in handling the complexities involved in formalising ManufacturingMethod at Meta and Individual levels of knowledge.

A2.2.1 Generic Core Concepts

```

:Use MCCO

:Prop Feature
:Inst Type
:sup Object
:name "Feature"
:rem "A Feature is anything having a particular attribute of interest"

:Prop Family
:Inst Type
:sup Object

```


:name "Family"

:rem "A Family is a group defined on the basis of a common criteria"

:Prop Criteria

:Inst Type

:sup Particular

:name "Criteria"

:rem "The concept Criteria specifies is used to specify the criteria for the definition of the Family. A Criteria can either be an object or an event."

:Prop Form

:Inst Type

:sup Object

:name ""

:Prop FormFeature

:Inst Type

:sup Feature

:name "FormFeature"

:Prop Material

:Inst Type

:sup Object

:name "material"

:Prop Resource

:Inst Type

:sup Object

:name "Resource"

:rem "Resource is a generic core concept in the MCCO"

:Prop Parameter

:Inst Type

:sup Object

:Prop Function

:Inst Type

:sup AttributeOfInterest

:sup Event

:name "Function"

:rem "Function is one of the generic core concepts that captures the functionality of entities"

A2.2.2 Generic Core Relations

:Rel hasAttributeOfInterest

:Inst BinaryRel

:Sig Feature Particular

```
:name "hasAttributeOfInterest"
:rem "The relation hasAttributeOfInterest relates the Particulars to the Features as their defining
attributes"
```

```
:Rel hasCriteria
:Inst BinaryRel
:Sig Family Criteria
:name "hasCriteria"
:rem "The relation hasCriteria helps define the criteria for a Family and its sub-concepts"
```

A2.2.3 Generic Core Axioms

```
;;;Axiom GC1 "Every feature has an Attribute of Interest"
```

```
(=> (Feature ?f)
      (exists(?AOI)
        (and (Particular ?AOI)
              (hasAttributeOfInterest ?f ?AOI))))
```

```
:IC hard "Every feature has an Attribute of Interest"
```

```
;Axiom for the concept Family
```

```
(=> (Family ?fam)
      (exists(?cri)
        (and (Criteria ?cri)
              (hasCriteria ?fam ?cri))))
```

```
:IC hard "The criteria for the Family may be defined"
```

;This axiom states that If there is a Family `?fam` then a Criteria `?cri` of that Family should exist. This Criteria should be related to the Family through relation hasCriteria. This implies that in order to assert a Family in the KB is Criteria has to be defined and related to it in accordance with the definition of Family."

```
;Axiom GC3 "If there is a FormFeature then there exists a Form"
```

```
(=> (FormFeature ?ffeature)
      (exists (?form)
        (and (Form ?form)
              (FormFeature ?ffeature)
              (hasAttributeOfInterest ?ffeature ?form))))
```

```
:IC hard "Eevery FormFeature has a form"
```

A2.3 Product Lifecycle Generic Level Formalisation

This section details the product lifecycle generic level core concepts, relations, functions and axioms. The product lifecycle generic level core concepts belonging to the *ManufacturingMethod* category are mentioned only in the section describing the formalisation of *ManufacturingMethod*. This is done to provide some ease in handling the complexities involved in formalising *ManufacturingMethod* at *Meta* and *Individual* levels of knowledge.

A2.3.1 Product Lifecycle Core Concepts

```
:Prop Part
:Inst Type
:sup Object
:name "Part"
```

```
:Prop Product
:Inst Type
:sup Object
```

```
;=====RealisedPart=====
```

```
:Prop RealisedPart
:Inst Type
:sup Object
:name "RealisedPart"
```

```
:Prop ManufacturedPart
:Inst Type
:sup RealisedPart
:name "ManufacturedPart"
```

```
:Prop ServicePart
:Inst Type
:sup RealisedPart
:name "ServicePart"
```

```
;=====PartFeature and PartFamily=====
```

```
:Prop PartFeature
:Inst Type
:sup FormFeature
:name "PartFeature"
```

```
:Prop PartFamily
:Inst Type
:sup Family
:name "PartFamily"
```

```
;=====PartVersion=====
```

```
:Prop PartVersion
:Inst Type
:sup Part
:name "PartVersion"
:rem "The PartVersion represents a version of a Part which captures the history of the Part"
```

```
:Prop PlannedPartVersion
:Inst Type
:sup PartVersion
```

```

;=====ManufacturingProcess=====
:Prop ManufacturingProcess
:Inst Type
:sup Event

;=====StandardFeature=====
:Prop StandardFeature
:Inst Type
:sup ProductFeature
:name "StandardFeature"
:rem "(StandardFeature ?stdfeature) Is a PartFeature that has been standardised across product design
and production domains"

;=====ManufacturingMethod=====
:Prop ManufacturingMethod
:Inst Type
:sup Event
:name " ManufacturingMethod "

;=====ManufacturingFacility=====
:Prop ManufacturingFacility
:Inst Type
:sup Object
:name "ManufacturinFacility"

:Prop Enterprise
:Inst Type
:sup ManufacturingFacility
:name "Enterprise"

:Prop Factory
:Inst Type
:sup ManufacturingFacility
:name "Factory"

:Prop Shop
:Inst Type
:sup ManufacturingFacility
:name "Shop"

:Prop Cell
:Inst Type
:sup ManufacturingFacility
:name "Cell"

:Prop Station

```

```

:Inst Type
:sup ManufacturingFacility
:name "Station"

;=====ManufacturingResource=====

:Prop ManufacturingResource
:Inst Type
:sup Resource
:name "ManufacturingResource"

:Prop HumanResource
:Inst Type
:sup ManufacturingResource
:name "HumanResource"
:rem "HumanResource is a sub concept of the main concept ManufacturingResource in MCCO"

:Prop Operator
:Inst Type
:sup HumanResource

;=====Miscellaneous Product Lifecycle Generic Core Concepts =====

:Prop Dimension
:Inst Type
:sup Object

:Prop LinearDimension
:Inst Type
:sup Dimension

:Prop AngularDimension
:Inst Type
:sup Dimension

```

A2.3.2 Product Lifecycle Core Relations

```

:Rel hasParameter
:Inst BinaryRel
:Sig Object Parameter
:name "hasParameter"

:Rel associatedTo
:Inst BinaryRel
:Sig PartFeature Part
:name "associatedTo"

:Rel hasValue
:Inst BinaryRel
:Sig Parameter Dimension

```

```
:Rel Holds
:Inst BinaryRel
:Sig Type Type
:Args "ManufacturingFacility" "ManufacturingResource"
```

```
:Rel hasCapabilityFor
:Inst BinaryRel
:Sig Type Type
:Args "ManufacturingFacility" "ManufacturingProcess"
```

```
:Rel Uses
:Inst BinaryRel
:Sig Type Type
:Args "ManufacturingProcess" "ManufacturingResource"
:Args "ProductionMethod" "ManufacturingFacility"
```

```
:Rel Produces
:Inst BinaryRel
:Sig ManufacturingFacility RealisedPart
:Args "ManufacturingFacility" "RealisedPart"
```

```
:Rel hasState
:Inst BinaryRel
:Sig RealisedPart PartState
```

```
:Rel holdsProductionResource
:Inst BinaryRel
:Sig Type Type
:Args "ManufacturingFacility" "ManufacturingResource"
```

```
:Rel hasDimension
:Inst TernaryRel
:Sig Object Parameter Dimension
:Args "Object" "Parameter" "Dimension"
```

A2.3.3 Product Lifecycle Core Functions

```
:Fun mm
:Inst UnaryFun
:Sig RealNumber -> LinearDimension
```

```
:Fun deg
:Inst UnaryFun
:Sig RealNumber -> AngularDimension
```

The next two functions are used to infer the DesignFeature and ProductionFeature when a StandardFeature is asserted in the KB. Because StandardFeature lays at the product lifecycle level, therefore, these functions are classified as ProductLifecycle level functions. These function are not the core functions and they only play a supportive role in building the axioms.

```
:Fun DesignFeaturefor
:Inst UnaryFun
:Sig StandardFeature -> DesignFeature
```

```
:Fun ProductionFeaturefor
:Inst UnaryFun
:Sig StandardFeature -> ProductionFeature
```

A2.3.4 Product Lifecycle Core Axioms

```
;Formalisation os the concept Form
;;Axiom: "Parameter(s) are needed to define a form"
(=> (Form ?form)
  (exists (?p)
    (and (Parameter ?p)
      (hasParameter ?form ?p))))
:IC hard "Parameter(s) of the form need to be defined"
```

```
;Formalisation of PartFeature
;;Axiom "A Part feature may have an associated Part
(=> (PartFeature ?Pfeature)
  (exists (?P)
    (and (Part ?P)
      (associatedTo ?Pfeature ?P))))
:IC soft "A Part feature may have an associated Part"
```

```
;Formalisation of PartFamily
(=> (PartFamily ?pf)
  (exists (?pt)
    (and (Part ?pt)
      (hasCriteria ?pf ?pt))))
:IC hard "The parametric Part <code> ?pt </code> may be defined for the PartFamily"
```

```
;Formalisation of StandardFeature
```

Both defining and controlling axioms as well as inference rules are needed for the heavyweight formalisation of semantics of StandardFeature.

Defining Axioms: Two defining axioms capture the relationship of standard feature with its attributes of interest.

1st defining axioms for standard feature captures its relation with the function and places it as an integrity constraint for the existence of standard feature.

1. Standard Feature Defining Axioms-1

```
;There has to exist function for a StandardFeature
(=> (StandardFeature ?stdfeature)
    (exists (?fun)
      (and (Function ?fun)
            (hasAttributeOfInterest ?stdfeature ?fun))))
:IC hard "There has to exist function for a StandardFeature"
```

2. Standard Feature Defining Axioms-2

;Axiom for formalising that there has to exist a ProductionMethod for every StandardFeature

```
( => (StandardFeature ?stdfeature)
  ( exists (?Productionmethod)
    (and (ProductionMethod ?Productionmethod)
          (hasAttributeOfInterest ?stdfeature ?Productionmethod))))
```

:IC hard "There has to exist a ProductionMethod for every StandardFeature"

The above two ICs capture the definition formally that for a StandardFeature to exist its required attributes of interests i.e. Function and ProductionMethod has to exist. This is understood by the system and this will constrain the user from asserting any StandardFeature with its requires attributes of Interest.

Standard feature Inference Rules: Since the standard feature is generic to both design and Production. It shares the attributes of design as well as manufacturing feature. Whenever a standard feature is populated in the KB it means that a manufacturing feature as well as a design feature are also populated through the inference rules. There one inference rule each for inferring a design feature and one for production feature.

1. Standard Feature Inference Axioms-1

The first rules infers a design feature from a standard feature whenever a standard feature fact is loaded. This rule infers a design feature from a loaded standard feature having same form and function as the standard feature.

```
(<= (and (DesignFeature (DesignFeaturefor ?stdfeature))
        (hasAttributeOfInterest (DesignFeaturefor ?stdfeature) ?function)
        (hasAttributeOfInterest (DesignFeaturefor ?stdfeature) ?form))
    (and (StandardFeature ?stdfeature)
        (Function ?function)
        (Form ?form)
        (hasAttributeOfInterest ?stdfeature ?function)
        (hasAttributeOfInterest ?stdfeature ?form)))
```


In simple English the above rule says that for a standard feature ?stdfeature which has a function ?function as an attribute of interest and a form ?form as an attribute of interest there is inferred a design feature 'DesignFeaturefor' which has same function ?function as an attribute of interest and a same form ?form as an attribute of interest.

2. Standard Feature controlling Axioms-1

The second rule infers a production feature from a standard feature whenever a standard feature fact is loaded. This rule infers a production feature from a loaded standard feature having same form and production method as the standard feature.

```
(<= (and (ProductionFeature (ProductionFeaturefor ?stdfeature))
(hasAttributeOfInterest (ProductionFeaturefor ?stdfeature) ?Productionmethod)
      (hasAttributeOfInterest (ProductionFeaturefor ?stdfeature) ?form))
  (and (StandardFeature ?stdfeature)
        (ProductionMethod ?Productionmethod) (Form ?form)
        (hasAttributeOfInterest ?stdfeature ?Productionmethod)
        (hasAttributeOfInterest ?stdfeature ?form))))
```

The above rule states that for a standard feature ?stdfeature which has a production method ?Productionmethod as an attribute of interest and a form ?form as an attribute of interest there is inferred a production feature 'ProductionFeaturefor' which has same production method ?Productionmethod as an attribute of interest and a same form ?form as an attribute of interest.

; There has to exist Function and ProductionMethod for every StandardFeature

```
(=> (StandardFeature ?stdfeature)
  (exists (?Productionmethod)
    (and (ProductionMethod ?Productionmethod)
          (hasAttributeOfInterest ?stdfeature ?Productionmethod))))
```

:IC hard "There has to exist Function and ProductionMethod for every StandardFeature"

;Axiom to formalise that a function should exist for a StandardFeature

```
(=> (StandardFeature ?stdfeature)
  (exists (?fun)
    (and (Function ?fun)
          (hasAttributeOfInterest ?stdfeature ?fun))))
```

:IC hard "There has to exist function for a StandardFeature"

Inference Rules for StandardFeature's formalisation

The below mentioned inference rules will infer a ProductionFeature and a DesignFeature whenever a StandardFeature fact is asserted in the KB.

```
(hasAttributeOfInterest (DesignFeaturefor ?stdfeature) ?function)
  (hasAttributeOfInterest (DesignFeaturefor ?stdfeature) ?form))
  (and (StandardFeature ?stdfeature)
```

```

(Function ?function)
(Form ?form)
(hasAttributeOfInterest ?stdfeature ?function)
(hasAttributeOfInterest ?stdfeature ?form)))

(<= (and (ProductionFeature (ProductionFeaturefor ?stdfeature))
  (hasAttributeOfInterest (ProductionFeaturefor ?stdfeature) ?Productionmethod)
  (hasAttributeOfInterest (ProductionFeaturefor ?stdfeature) ?form))
  (and (StandardFeature ?stdfeature)
    (ProductionMethod ?Productionmethod)
    (Form ?form)
    (hasAttributeOfInterest ?stdfeature ?Productionmethod)
    (hasAttributeOfInterest ?stdfeature ?form))))

```

A2.4 Design and Production Level Formalisation

A2.4.1 Production Level Formalisation

This section details the production level core concepts, relations, functions and axioms. The production level core concepts belonging to the *ManufacturingMethod* category are mentioned only in the section describing the formalisation of *ManufacturingMethod*. This is done to provide some ease in handling the complexities involved in formalising *ManufacturingMethod* at *Meta* and *Individual* levels of knowledge.

A2.4.1.1 Production Core Concepts

```
:Prop ProductionFeature
```

```
:Inst Type
```

```
:sup PartFeature
```

```
:name ""
```

```
:Prop ProductionPartFamily
```

```
:Inst Type
```

```
:sup PartFamily
```

```
:name "ProductionPartFamily"
```

```
:Prop Fixture
```

```
:Inst Type
```

```
:sup ManufacturingResource
```

```
:name "HumanResource"
```

```
:Prop ToolHolding
```

```
:Inst Type
```

```
:sup Fixture
```

```
:Prop Workpiece
```

```
:Inst Type
```

```
:sup ManufacturingResource
```

:Prop MachineTool
:Inst Type
:sup ManufacturingResource
:rem "MachineTool is a sub concept of the main concept ManufacturingResource in the MCCO"

:Prop CuttingTool
:Inst Type
:sup ManufacturingResource
:name "CuttingTool"
:rem "CuttingTool is a sub concept of the main concept ManufacturingResource in the MCCO"

:Prop ProductionPartVersion
:Inst Type
:sup PlannedPartVersion

:Prop ProductionPartState
:Inst MaterialRole
:sup Type

:Prop PartInSpecification
:Inst Type
:sup ProductionPartState

:Prop RejectedPart
:Inst Type
:sup ProductionPartState

:Prop Concession
:Inst Type
:sup ProductionPartState

:Prop Prototype
:Inst Type
:sup ProductionPartState

:Prop WorkInProgress
:Inst Type
:sup ProductionPartState

A2.4.1.2 Production Core Relations

:Rel canMachine
:Inst BinaryRel
:Sig CuttingTool Object
:Rel CannotbeUsed

:Inst UnaryRel
:Sig CuttingTool

:Rel hasState
:Inst BinaryRel
:Sig ProductionPart ProductionPartState

A2.4.1.3 Production Core Axioms

;Axiom for Formalisation of the concept ProductionFeature

```
(=> (ProductionFeature ?Productionf)
      (exists (?Productionmeth)
                (and (ProductionMethod ?Productionmeth)
                     (hasAttributeOfInterest ?Productionf ?Productionmeth))))
```

:IC hard "Production method may be defined for a Productionfeature"

```
(=> (and (ProductionFeature ?Productionf)(Function ?function))
      (not (hasAttributeOfInterest ?Productionf ?function)))
```

:IC hard "Function does not apply to a production feature"

:Rel hasToolWidthAndClearanceValue
:Inst BinaryRel
:Inst IntensionalRel
:Sig CuttingTool Dimension

:Rel CannotbeUsed
:Inst UnaryRel
:Sig CuttingTool

A2.4.2 Design Level Formalisation

This section details the design level core concepts, relations, functions and axioms.

A2.4.2.1 Design Core Concepts

:Prop DesignFeature
:Inst Type
:sup ProductFeature
:name "DesignFeature"

:Prop DesignPartFamily
:Inst Type
:sup PartFamily
:name "DesignPartFamily"

:Prop DesignPartVersion
:Inst Type
:sup PlannedPartVersion

A2.4.2.2 Design Core Relations and Functions

The relations and functions defined at the generic core level and product lifecycle generic core level are sufficient to define the semantics of design core concepts. Examples of these relations are *hasAttributeOfInterest* and *hasCriteria*.

A2.4.2.3 Design Core Axioms

```
;"function exists for every DesignFeature"
```

```
(=> (DesignFeature ?df)
      (exists (?function)
              (and (Function ?function)
                    (hasAttributeOfInterest ?df ?function))))
```

```
:IC hard "Function (AttributeOfInterest) for DesignFeature not defined"
```

```
;"A design feature cannot be defined with ProductionMethod as its Attribute of Interest"
```

```
(=>(and (DesignFeature ?df) (ProductionMethod ?ProductionMethod))
      (not(hasAttributeOfInterest ?df ?ProductionMethod)))
```

```
:IC hard "ProductionMethod does not apply to DesignFeature"
```

```
;(=> (DesignFeature ?df)
      ;(not(exists (?ProductionMethod)
                  ;(and (ProductionMethod ?ProductionMethod)
                        ; (hasAttributeOfInterest ?df ?ProductionMethod))))))
:IC hard "ProductionMethod does not apply to DesignFeature"
```

A2.5 ProductionMethod Formalisation for knowledge capture and reasoning at Meta and Individual levels of knowledge

```
; Parent concepts of clajects instantiated from powertypes
```

```
:Prop FeatureProductionMethod
```

```
:Inst Type
```

```
:sup ProductionMethod
```

```
:Prop Stage
```

```
:Inst Type
```

```
:sup ProductionMethod
```

```
:Prop Setup
```

```
:Inst Type
```

```
:sup ProductionMethod
```

```
:Prop ProcessPlan
```

```
:Inst Type
```

```
:sup ProductionMethod
```

```
:Prop Operation
```

```
:Inst Type
```

:sup ProductionMethod

:Prop Step

:Inst Type

:sup ProductionMethod

; Powertypes or MetaProperties

:Prop MetaFeatureProductionMethod

:Inst MetaProperty

:sup Type

:metaPropFor FeatureProductionMethod

:Prop MetaStage

:Inst MetaProperty

:sup Type

:metaPropFor Stage

:Prop MetaSetup

:Inst MetaProperty

:sup Type

:metaPropFor Setup

:Prop PartFamilyProductionMethod

:Inst MetaProperty

:sup Type

:metaPropFor ProcessPlan

:Prop MetaOperation

:Inst MetaProperty

:sup Type

:metaPropFor Operation

:Prop MetaStep

:Inst MetaProperty

:sup Type

:metaPropFor Step

; Clabjects instantiated from Powertypes of MetaProperties

; Clabjects of MetaFeatureProductionMethod / sub concepts of FeatureProductionMethod

:Prop CircumGrooveProductionMethod

:Inst MetaFeatureProductionMethod

:sup FeatureProductionMethod

```

:Prop HoleProductionMethod
:Inst MetaFeatureProductionMethod
:sup FeatureProductionMethod

```

```

:Prop DiaphragmProductionMethod
:Inst MetaFeatureProductionMethod
:sup FeatureProductionMethod

```

; Clbjects of MetaStage / sub concepts of Stage

```

:Prop TurningStage
:Inst MetaStage
:sup Stage

```

```

:Prop MillingStage
:Inst MetaStage
:sup Stage

```

```

:Prop GrindingStage
:Inst MetaStage
:sup Stage

```

; clbjects of MetaStep / sub concepts of Steps

```

:Prop RoughTurningtep
:Inst MetaStep
:sup Step

```

```

:Prop FinishTurningtep
:Inst MetaStep
:sup Step

```

```

:Prop RoughMillingtep
:Inst MetaStep
:sup Step

```

```

:Prop RoughGrindingStep
:Inst MetaStep
:sup Step

```

; Clbjects of MetaSetup / subconcepts of Setup

```

:Prop TurningSetup
:Inst MetaSetup
:sup Setup

```

```

:Prop TwinTurningSetup
:Inst MetaSetup
:sup Setup

```

```

:Prop RoughTurningSetup
:Inst MetaSetup
:sup TurningSetup

```

```

:Prop FinishTurningSetup
:Inst MetaSetup
:sup TurningSetup

```

```

:Prop MillingSetup
:Inst MetaSetup
:sup Setup

```

```

:Prop RoughMillingSetup
:Inst MetaSetup
:sup MillingSetup

```

```

:Prop FinishMillingSetup
:Inst MetaSetup
:sup MillingSetup

```

```

:Prop GrindingSetup
:Inst MetaSetup
:sup Setup

```

; clabjects of PartFamilyProductionMethod / sub-concepts concepts ProcessPlan

```

:Prop StandAloneDiscProcessPlan
:Inst PartFamilyProductionMethod
:sup ProcessPlan

```

```

:Prop ProjectedDiscProcessPlan
:Inst PartFamilyProductionMethod
:sup ProcessPlan

```

; Clabjects of MetaOperation / sub concepts of Operation

```

:Prop TurningOperation
:Inst MetaOperation
:sup Operation

```

```

:Prop TwinTurningOperation

```



```

:Inst MetaOperation
:sup TurningOperation

:Prop RoughTurningOperation
:Inst MetaOperation
:sup TurningOperation

:Prop FinishTurningOperation
:Inst MetaOperation
:sup TurningOperation

:Prop MillingOperation
:Inst MetaOperation
:sup Operation

:Prop RoughMillingOperation
:Inst MetaOperation
:sup TurningOperation

:Prop FinishMillingOperation
:Inst MetaOperation
:sup MillingOperation

:Prop DrillingOperation
:Inst MetaOperation
:sup Operation

:Prop GrindingOperation
:Inst MetaOperation
:sup Operation

```

; Relations that may apply to the instances of metaconcepts i.e. concepts and to the instances of concepts
i.e. individuals

```

:Rel hasStage
:Inst BinaryRel
:Sig Top Top

```

```

:Rel hasOperation
:Inst BinaryRel
:Sig Top Top

```

```

:Rel hasSetup
:Inst BinaryRel
:Sig Top Top

```

```
:Rel hasStep
:Inst BinaryRel
:Sig Top Top
```

```
:Rel minPrecedes
:Inst TernaryRel
:Sig Top Top Top
```

```
:Rel canBeManufacturedIn
:Inst QuaternaryRel
:Sig Top Top Top Top
```

```
:Rel canBeAccommodatedIn
:Inst BinaryRel
:Sig Top Top
```

; Integrity constraints to confine the domain and ranges of relevant relations whose signatures involve Top

```
(=> (hasStage ?x ?y)
    (or (and (MetaFeatureProductionMethod ?x)
              (MetaStage ?y))
        (and (FeatureProductionMethod ?x)
              (Stage ?y))))
```

:IC hard "The relation hasStage holds between instances of MetaFeatureProductionMethod and MetaStage or FeatureProductionMethod and Stage respectively."

```
(=> (hasOperation ?x ?y)
    (or (and (PartFamilyProductionMethod ?x)
              (MetaOperation ?y))
        (and (ProcessPlan ?x)
              (Operation ?y))))
```

:IC hard "The relation hasOperation holds between instances of PartFamilyProductionMethod and MetaOperation or ProcessPlan and Operation respectively."

```
(=> (hasSetup ?x ?y)
    (or (and (MetaStage ?x)
              (MetaSetup ?y))
        (and (Stage ?x)
              (Setup ?y))
        (and (MetaOperation ?x)
              (MetaSetup ?y))
        (and (Operation ?x)
              (Setup ?y))))
```

:IC hard "The relation hasSetup holds between concepts of Stage and Setup and their instances or concepts of Operation and Setup and their instances respectively."

```
(=> (hasStep ?x ?y)
      (or (and (MetaStage ?x)
                (MetaStep ?y))
          (and (Stage ?x)
                (Step ?y))
          (and (MetaOperation ?x)
                (MetaStep ?y))
          (and (Operation ?x)
                (Step ?y))))
```

:IC hard "The relation hasSetup holds between concepts of Stage and Setup and their instances or concepts of Operation and Setup and their instances respectively."

```
(=> (minPrecedes ?x ?y ?mm)
      (or (and (MetaStage ?x)
                (MetaStage ?y)
                (MetaFeatureProductionMethod ?mm))
          (and (Stage ?x)
                (Stage ?y)
                (FeatureProductionMethod ?mm))
          (and (MetaOperation ?x)
                (MetaOperation ?y)
                (PartFamilyProductionMethod ?mm))
          (and (Operation ?x)
                (Operation ?y)
                (ProcessPlan ?mm))
          (and (Step ?x)
                (Step ?y)
                (Stage ?mm))
          (and (MetaStep ?x)
                (MetaStep ?y)
                (MetaStage ?mm))
          (and (Step ?x)
                (Step ?y)
                (Operation ?mm))
          (and (MetaStep ?x)
                (MetaStep ?y)
                (MetaOperation ?mm))))
```

:IC hard "The relation minPrecedes holds between concepts of Stage and their instances for corresponding concepts of MetaFeatureProductionMethod and their instances or between concepts of Operation and their instances for corresponding concepts of Process Plan and their instances respectively or between concepts of step and their instances and their corresponding stages and their instances respectively or between concepts of step and their instances and their corresponding Operations and their instances respectively."

```
(=> (canBeManufacturedIn ?a ?b ?x ?y)
```

```

(or (and (MetaStage ?a)
         (MetaFeatureProductionMethod ?b)
         (MetaOperation ?x)
         (PartFamilyProductionMethod ?y))
    (and (Stage ?a)
         (FeatureProductionMethod ?b)
         (Operation ?x)
         (ProcessPlan ?y))))

```

:IC hard "The relation canBeManufacturedIn holds between concepts of Stage and their instances, concepts of FeatureProductionMethod and their instances, concepts of Operation and their instances, and concepts of ProcessPlan and their instances."

```

(=> (canBeAccommodatedIn ?x ?y)
    (or (and (MetaFeatureProductionMethod ?x)
             (PartFamilyProductionMethod ?y))
        (and (FeatureProductionMethod ?x)
             (ProcessPlan ?y))))

```

:IC hard "The relation canBeManufacturedIn holds between concepts of FeatureProductionMethod and their instances and concepts of ProcessPlan and their instances."

;;; Defining the linear ordering semantics of minPrecedes

```

(=> (and (Top ?arg1)
        (Top ?arg2)
        (Top ?arg3)
        (minPrecedes ?arg1 ?arg2 ?arg3))
    (not (= ?arg1 ?arg2)))

```

:IC hard "minPrecedes is Irreflexive."

```

(=> (and (Top ?arg1)
        (Top ?arg2)
        (Top ?arg3)
        (minPrecedes ?arg1 ?arg2 ?arg3))
    (not (minPrecedes ?arg2 ?arg1 ?arg3)))

```

:IC hard "minPrecedes is Antisymmetric."

```

(=> (and (Top ?arg1)
        (Top ?arg2)
        (Top ?arg3)
        (Top ?arg4)
        (minPrecedes ?arg1 ?arg2 ?arg3)
        (minPrecedes ?arg2 ?arg4 ?arg3))
    (minPrecedes ?arg1 ?arg4 ?arg3))

```

:rem "minPrecedes is Transitive."

;;; Inference rule to establish concepts of Stage that can be machined in concepts of Operation

```
(<= (canBeManufacturedIn ?st ?fmm ?op ?pfmm)
  (and (MetaFeatureProductionMethod ?fmm)
    (MetaStage ?st)
    (MetaSetup ?set1)
    (hasStage ?fmm ?st)
    (hasSetup ?st ?set1)
    (PartFamilyProductionMethod ?pfmm)
    (MetaOperation ?op)
    (MetaSetup ?set2)
    (hasSetup ?op ?set2)
    (= ?set1 ?set2))))
```

:rem "The stage class <code>?st</code> of the feature ProductionMethodclass <code>?fmm</code> can be machined within the operation class <code>?op</code> of the part family ProductionMethodclass <code>?pfmm</code>, since the stage class and the operation class share the same setup class."

; Inference rule to establish instances of Stage that can be machined in instances of Operation

```
(<= (canBeManufacturedIn ?st ?fmm ?op ?pfmm)
  (and (FeatureProductionMethod ?fmm)
    (Stage ?st)
    (Setup ?set1)
    (hasStage ?fmm ?st)
    (hasSetup ?st ?set1)
    (ProcessPlan ?pfmm)
    (Operation ?op)
    (Setup ?set2)
    (hasSetup ?op ?set2)
    (= ?set1 ?set2))))
```

:rem "The stage instance <code>?st</code> of the feature ProductionMethodinstance <code>?fmm</code> can be machined within the operation instance <code>?op</code> of the process plan instance <code>?pfmm</code>, since the stage instance and the operation instance share the same setup instance."

; Inference rules to establish clabjects of MetaFeatureProductionMethod that can be accommodated in clabjects of PartFamilyProductionMethod

```
:Rel canBeAccommodatedIn_False
:Inst BinaryRel
:Inst IntensionalRel
:Sig Top Top
```

```
(<= (canBeAccommodatedIn_False ?fmm ?pfmm)
  (and (MetaStage ?st1)
```

```

        (MetaStage ?st2)
        (minPrecedes ?st2 ?st1 ?fmm)
        (MetaOperation ?op1)
        (MetaOperation ?op2)
        (minPrecedes ?op1 ?op2 ?pfmm)
        (canBeManufacturedIn ?st1 ?fmm ?op1 ?pfmm)
        (canBeManufacturedIn ?st2 ?fmm ?op2 ?pfmm)))

(<= (canBeAccommodatedIn ?fmm ?pfmm)
    (and (MetaStage ?st)
          (MetaOperation ?op)
          (canBeManufacturedIn ?st ?fmm ?op ?pfmm)
          (not (canBeAccommodatedIn_False ?fmm ?pfmm))))

:rem "The FeatureProductionMethod <code>?fmm</code> can be accommodated in the process plan
class <code>?pfmm</code>."

;;; Inference rules to establish intances of FeatureProductionMethod that can be accommodated in
instances of ProcessPlan

(<= (canBeAccommodatedIn_False ?fmm ?pfmm)
    (and (Stage ?st1)
          (Stage ?st2)
          (minPrecedes ?st2 ?st1 ?fmm)
          (Operation ?op1)
          (Operation ?op2)
          (minPrecedes ?op1 ?op2 ?pfmm)
          (canBeManufacturedIn ?st1 ?fmm ?op1 ?pfmm)
          (canBeManufacturedIn ?st2 ?fmm ?op2 ?pfmm)))

(<= (canBeAccommodatedIn ?fmm ?pfmm)
    (and (Stage ?st)
          (Operation ?op)
          (canBeManufacturedIn ?st ?fmm ?op ?pfmm)
          (not (canBeAccommodatedIn_False ?fmm ?pfmm))))

:rem "The feature ProductionMethodinstance <code>?fmm</code> can be accommodated in the process
plan instance <code>?pfmm</code>."

;===== Axioms for Incorporating Step in the ProductionMethod =====

;( => (Stage ?stg)
;      (exists (?stp)
;      (and (Step ?stp)
;            (hasStep ?stg ?stp))))

;:IC hard "Eevery Stage should consist of atleast one step"

;( => (MetaStage ?stg)

```

```

;      (exists (?stp)
;      (and      (MetaStep ?stp)
;      (hasStep ?stg ?stp))))
;:IC hard "Eevery MetaStage should consist of atleast one MetaStep"

```

```

;( => (Operation ?op)
;      (exists (?stp)
;      (and      (Step ?stp)
;      (hasStep ?op ?stp))))
;:IC hard "Eevery Operation should consist of atleast one step"

```

```

;( => (MetaOperation ?op)
;      (exists (?stp)
;      (and      (MetaStep ?stp)
;      (hasStep ?op ?stp))))
;:IC hard "Eevery MetaOperation should consist of atleast one MetaStep"

```

```

;=====Facts madeup of Clabjects to be loaded with the ontology=====
(MCCO.hasSetup MCCO.TurningStage MCCO.TurningSetup)
(MCCO.hasSetup MCCO.MillingStage MCCO.MillingSetup)

```

```

(MCCO.hasSetup MCCO.TurningOperation MCCO.TurningSetup)
(MCCO.hasSetup MCCO.TwinTurningOperation MCCO.TwinTurningSetup)
(MCCO.hasSetup MCCO.RoughTurningOperation MCCO.RoughTurningSetup)
(MCCO.hasSetup MCCO.MillingOperation MCCO.MillingSetup)
(MCCO.hasSetup MCCO.RoughMillingOperation MCCO.RoughMillingSetup)
(MCCO.hasSetup MCCO.FinishMillingOperation MCCO.FinishMillingSetup)

```

```

; Feature ProductionMethod for Diaphragm
(MCCO.hasStage MCCO.DiaphragmProductionMethod MCCO.TurningStage)
(MCCO.hasStage MCCO.DiaphragmProductionMethod MCCO.MillingStage)
(MCCO.minPrecedes MCCO.MillingStage MCCO.TurningStage MCCO.DiaphragmProductionMethod)

```

```

; Feature ProductionMethod for circum groove
(MCCO.hasStage MCCO.CircumGrooveProductionMethod MCCO.TurningStage)
(MCCO.hasStage MCCO.CircumGrooveProductionMethod MCCO.MillingStage)
(MCCO.minPrecedes MCCO.TurningStage MCCO.MillingStage MCCO.CircumGrooveProductionMethod)

```

```

; Part Family ProductionMethod for StandAloneDisc
(MCCO.hasOperation MCCO.StandAloneDiscProcessPlan MCCO.TurningOperation)
(MCCO.hasOperation MCCO.StandAloneDiscProcessPlan MCCO.MillingOperation)
(MCCO.minPrecedes MCCO.TwinTurningOperation MCCO.TurningOperation
MCCO.StandAloneDiscProcessPlan)
(MCCO.minPrecedes MCCO.TurningOperation MCCO.MillingOperation
MCCO.StandAloneDiscProcessPlan)

```

```

; Part Family ProductionMethod for ProjectedDisc
(MCCO.hasOperation MCCO.ProjectedDiscProcessPlan MCCO.TurningOperation)
(MCCO.hasOperation MCCO.ProjectedDiscProcessPlan MCCO.MillingOperation)
(MCCO.minPrecedes MCCO.RoughTurningOperation MCCO.FinishTurningOperation
MCCO.ProjectedDiscProcessPlan)
(MCCO.minPrecedes MCCO.FinishTurningOperation MCCO.RoughMillingOperation
MCCO.ProjectedDiscProcessPlan)
(MCCO.minPrecedes MCCO.RoughMillingOperation MCCO.FinishMillingOperation
MCCO.ProjectedDiscProcessPlan)

```

A2.6 MCCO Extensions

The MCCO extensions are provided to show the possible extension of the MCCO in various direction to include core concepts from several different modules of manufacturing. These extensions are inline with the discussion reported in chapter 9 of this thesis.

A2.6.1 ManufacturingResource Extension

;===== Lightweight formalisation fo Manufacturing Resource Extension=====

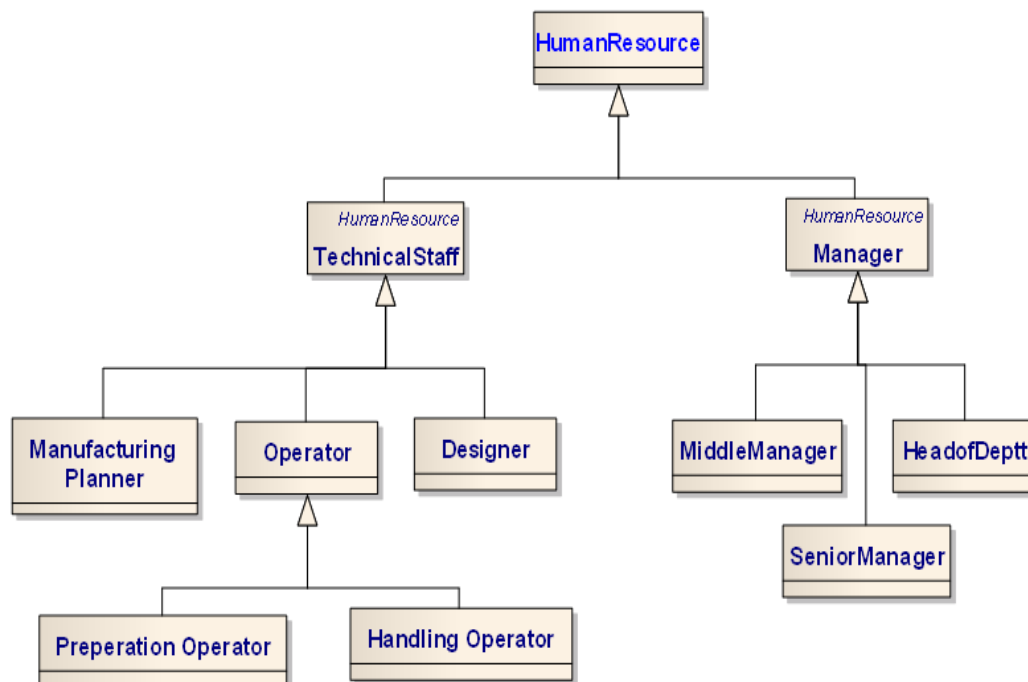


Figure A2.1 Lightweight formalisation fo Manufacturing Resource Extension

;=====Heavyweight formalisation fo Manufacturing Resource Extension =====

:Prop HumanResource

:Inst Type

:sup Resource

:name "HumanResource"

:Prop TechnicalStaff

:Inst Type

:sup HumanResource

:Prop TechnicalStaff

:Inst Type

:sup HumanResource

:Prop Operator

:Inst Type

:sup HumanResource

:sup ManufacturingResource

:Prop HandlingOperator

:Inst Type

:sup Operator

:Prop PreperationOperator

:Inst Type

:sup Operator

;===== ; Lightweight formalisation fo Fixture Core Concept =====

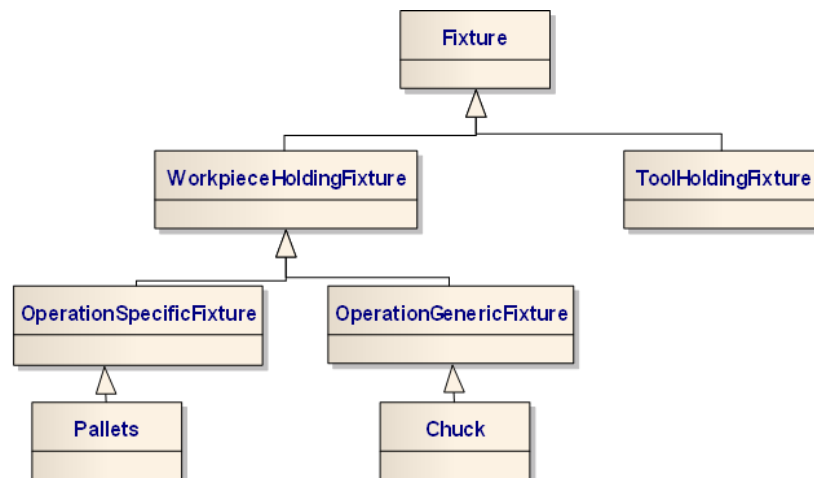


Figure A2.2: Lightweight Formalisation of extension of Fixture core concept

;===== Heavyweight formalisation fo Fixture Core Concept =====

:Prop Fixture

:Inst Type

:sup ManufacturingResource

:name "Fixture"

:Prop ToolHoldingFixture

:Inst Type

:sup Fixture

:Prop WorkPieceHoldingFixture

:Inst Type

:sup Fixture

:Prop OperationGenericFixture

:Inst Type

:sup WorkPieceHoldingFixture

:Prop Chuck

:Inst Type

:sup OperationGenericFixture

:Prop TSlotTable

:Inst Type

:sup OperationGenericFixture

:Prop OperationSpecificFixture

:Inst Type

:sup WorkPieceHoldingFixture

:Prop Pallet

:Inst Type

:sup OperationSpecificFixture

;===== Lightweight formalisation fo MachineTool Extension =====

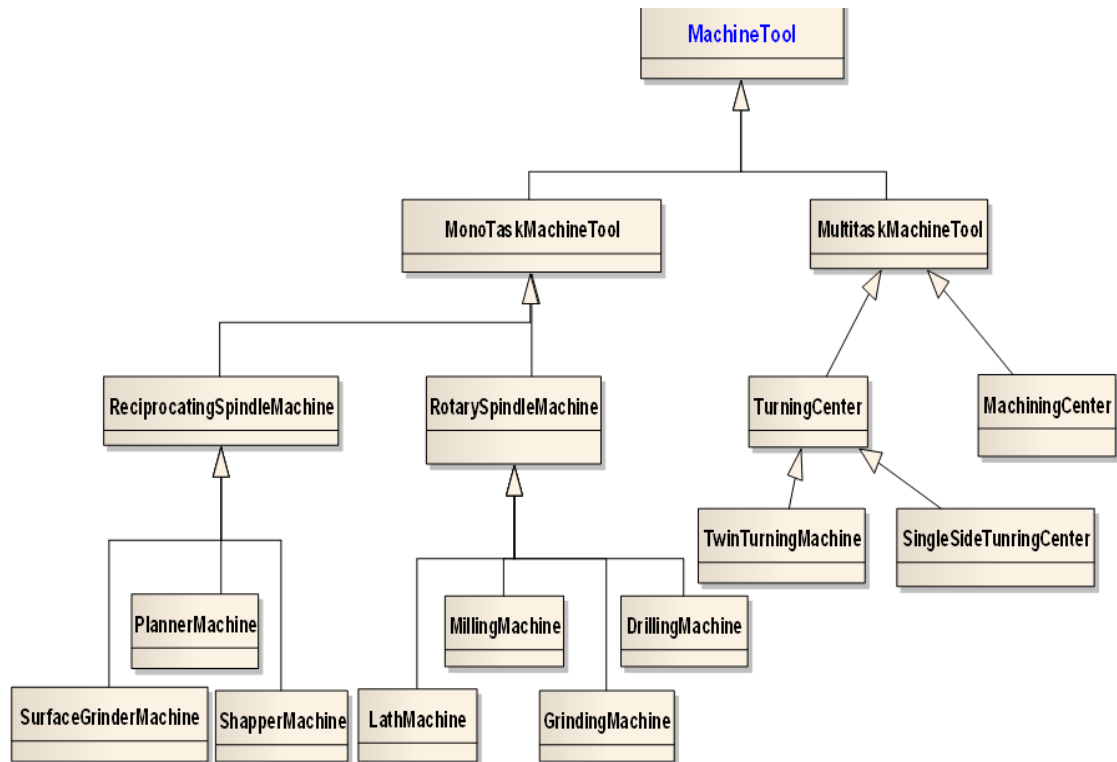


Figure A2.3: Lightweight Formalisation of extension of MachineTool core concept

;===== Heavyweight formalisation fo Fixture Core Concept =====

:Prop MachineTool

:Inst Type

:sup ManufacturingResource

:Prop MonoTaskMachineTool

:Inst Type

:sup MachineTool

:Prop ReciprocatingSpindleMachine

:Inst Type

:sup MonoTaskMachineTool

:Prop SurfaceGrinder

:Inst Type

:sup ReciprocatingSpindleMachine

:Prop PlannerMachine

:Inst Type

:sup ReciprocatingSpindleMachine

:Prop ShaperMachine

:Inst Type

:sup ReciprocatingSpindleMachine

:Prop RotarySpindleMachine

:Inst Type

:sup MonoTaskMachineTool

:Prop MonoTaskMachineTool

:Inst Type

:sup MachineTool

:Prop MillingMachine

:Inst Type

:sup MonoTaskMachineTool

:Prop LatheMachine

:Inst Type

:sup MonoTaskMachineTool

:Prop DrillingMachine

:Inst Type

:sup MonoTaskMachineTool

```

:Prop GrindingMachine
:Inst Type
:sup MonoTaskMachineTool

```

```

:Prop CuttingTool
:Inst Type
:sup ManufacturingResource

```

A2.6.2 ManufacturingProcess Extension

```

;====Lightweight formalisation of extension of ManufacturingProcess Extension

```

```

;====;=====Heavyweight formalisation of extension of ManufacturingProcess Extension ;=====

```

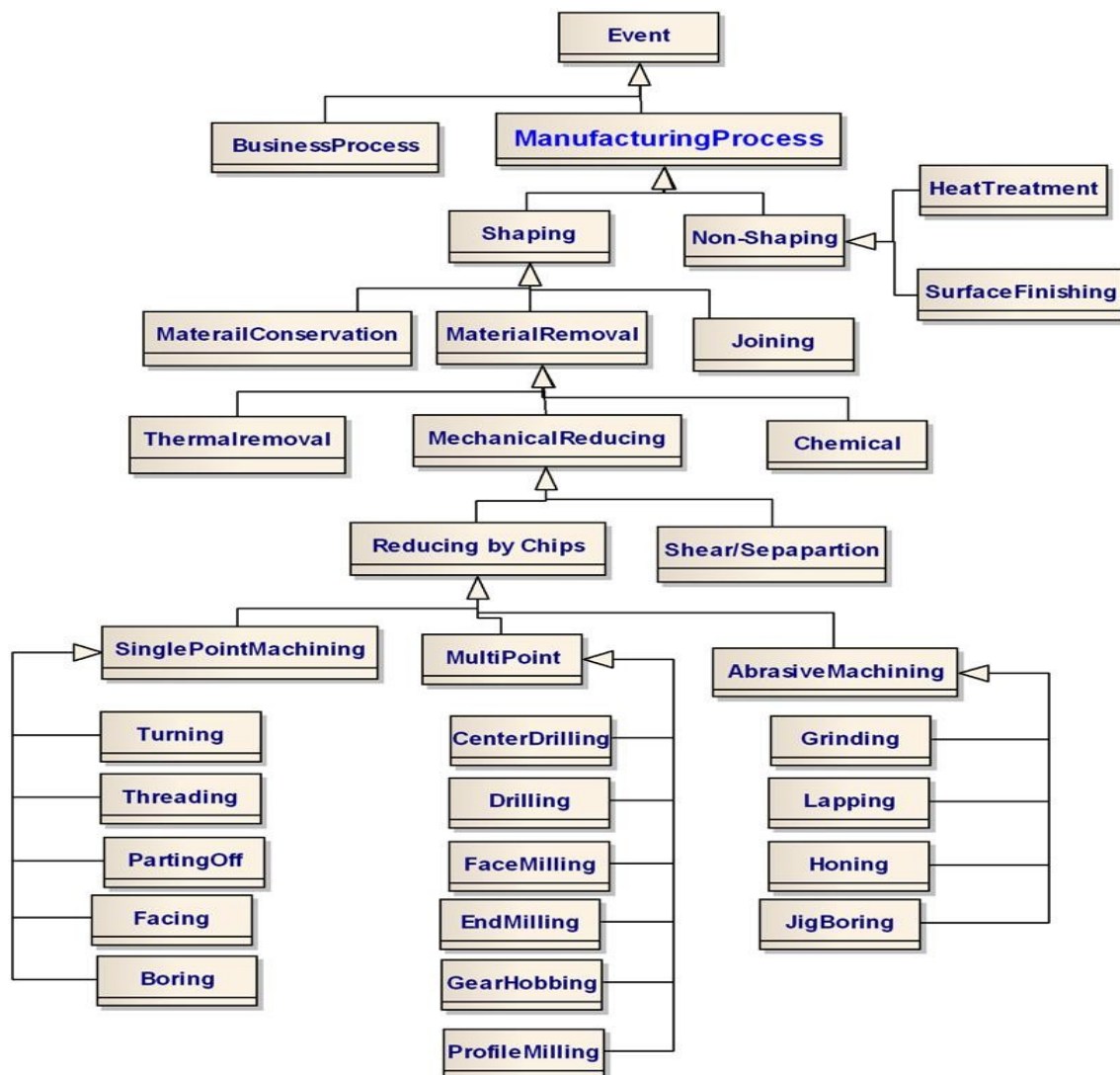


Figure A2.4: lightweight formalisation of ManufacturingProcess(Feng and Song 2003; Todd 1994)

```

:Use MCCO

```

```

:Prop ManufacturingProcess
:Inst Type
:sup Event

```

:Prop Shaping

:Inst Type

:sup ManufacturingProcess

:Prop MassReducing

:Inst Type

:sup Shaping

:Prop MechReducing

:Inst Type

:sup MassReducing

:Prop Chipremoval

:Inst Type

:sup MechReducing

:Prop Shearing

:Inst Type

:sup MassReducing

:Prop ThermalReducing

:Inst Type

:sup MassReducing

:Prop ChemReducing

:Inst Type

:sup MassReducing

:Prop MassConserving

:Inst Type

:sup Shaping

:Prop Joining

:Inst Type

:sup Shaping

:Prop NonShaping

:Inst Type

:sup ManufacturingProcess

A2.7 Specific Product Design and Production Ontologies

A2.7.1 Design specific Ontology 'Aero Engine Disc Design Ontology'

;===== Ontology developed from the MCCO =====

;Aero Engine Disc Design Ontology is an application specific ontology that is developed using the MCCO's semantic base. The Aero Engine Disc Design Ontology, however, has its own concepts with its own required semantics that does not violate the semantics provided by the MCCO. Therefore, the MCCO provides the semantic integrity and consistency to support interoperability of this ontology and its KB across to production.

:Use Design

:Prop DiscDesignFeature

:Inst Type

:sup MCCO.DesignFeature

:name "AeroEngineDiscDesignFeature"

:rem "DiscDesignFeature reoresents the design features in for aero engine discs"

:Prop StressReductionFeature

:Inst Type

:sup DiscDesignFeature

:Prop RimDiaphragmInterface

:Inst Type

:sup StressReductionFeature

:Prop DiaphragmCobInterface

:Inst Type

:sup StressReductionFeature

:Prop DefenderSlot

:Inst Type

:sup StressReductionFeature

:Prop BalancingFeature

:Inst Type

:sup DiscDesignFeature

:Prop BalanceLand

:Inst Type

:sup BalancingFeature

:Prop Holding&LoactingFeature

:Inst Type

:sup DiscDesignFeature

:Prop Circumference

:Inst Type

:sup Holding&LoactingFeature

:Prop CircumferentialGroove

:Inst Type

:sup Holding&LoactingFeature

:Prop LoadingSlot

:Inst Type

:sup Holding&LoactingFeature

:Prop LockingSlot

:Inst Type

:sup Holding&LoactingFeature

:Prop JoiningFeature

:Inst Type

:sup DiscDesignFeature

:Prop Diaphragm

:Inst Type

:sup JoiningFeature

:Prop SpigotEdge

:Inst Type

:sup JoiningFeature

:Prop BoltHole

:Inst Type

:sup JoiningFeature

:Prop CoolingFeature

:Inst Type

:sup DiscDesignFeature

:Prop Cob

:Inst Type

:sup CoolingFeature

A2.7.2 Production specific Ontology ‘Aero Engine Disc Production Ontology’

;=====Ontology developed from the MCCO=====

;Aero Engine Disc Production Ontology is an application specific ontology that is developed using the MCCO’s semantic base. The Aero Engine Disc Production Ontology, however, has its own concepts with its own required semantics that does not violate the semantics provided by the MCCO. Therefore, the MCCO provides the semantic integrity and consistency to support interoperability of this ontology and its KB across to product design.

:Use Production

:Prop DiscProductionFeature

:Inst Type

:sup MCCO.ProductionFeature

:name "DiscProductionFeature"

:rem "DiscProductionFeature represents the areas of the aero engine disc defined with respect to their methods of production"

:Prop TurningFeature

:Inst Type

:sup DiscProductionFeature

:Prop Rim

:Inst Type

:sup TurningFeature

:Prop GrindingFeature

:Inst Type

:sup DiscProductionFeature

:Prop TwinTurningFeature

:Inst Type

:sup DiscProductionFeature

:Prop Hub

:Inst Type

:sup TwinTurningFeature

:Prop WebProfile

:Inst Type

:sup TwinTurningFeature

:Prop DrillingFeature

:Inst Type

:sup DiscProductionFeature


```

:Prop WebHole
:Inst Type
:sup DrillingFeature

:Prop MillingFeature
:Inst Type
:sup DiscProductionFeature

:Prop RimSlot
:Inst Type
:sup MillingFeature

:Prop Groove
:Inst Type
:sup Form

:Prop NeckWidth
:Inst Type
:sup Parameter

:Prop GrooveAngle
:Inst Type
:sup Parameter

:Prop Hole
:Inst Type
:sup Parameter

;====Relations for capturing production knowledg about Rim and Webprofile machining =====

:Rel cuttingToolhasSCL
:Inst BinaryRel
:Sig MCCO.CuttingTool MCCO.LinearDimension

:Prop OuterRadius
:Inst Type
:sup MCCO.Parameter

:Prop InnerRadius
:Inst Type
:sup MCCO.Parameter

;Rule to Infer R+r

:Rel hasRplusValue

```

```
:Inst BinaryRel
:Inst IntensionalRel
:Sig WebProfile MCCO.LinearDimension
```

```
;=====Axioms to capture the production knowledg about machining of Rim =====
```

```
;Infer the sum of the tool width and the tool clearance values for a given tool
```

```
(=> (and (MCCO.CuttingTool ?ct)
  (MCCO.Parameter ?toolclearance)
  (MCCO.Parameter ?toolwidth)
  (/= ?toolclearance ?tooldia) ;These two variables need to be disambiguated.
  (MCCO.hasParameter ?ct ?toolclearance)
  (MCCO.hasParameter ?ct ?tooldia)
  (MCCO.hasValue ?toolclearance (MCCO.mm ?real1))
  (MCCO.hasValue ?tooldia (MCCO.mm ?real2))
  (numPlus ?real1 ?real2 ?real3))
  (MCCO.hasToolDiaAndClearanceValue ?ct (MCCO.mm ?real3)))
```

```
;Then, we construct the IC to only work on the inferred value from the previous inference rule:
```

```
(=> (and (Production.Rim ?rim)
  (MCCO.Form ?groove)
  (MCCO.hasAttributeOfInterest ?rim ?groove)
  (MCCO.NeckWidth ?nw)
  (MCCO.hasParameter ?groove ?nw)
  (MCCO.hasValue ?nw (MCCO.mm ?real1))
  (MCCO.CuttingTool ?ct)
  (MCCO.hasToolDiaAndClearanceValue ?ct (MCCO.mm ?real2)))
  (gteNum ?real1 ?real2))
```

```
:IC hard "The NeckWidth of Rim should be greater than diameter and clearance values of the CuttingTool
for the Rim to be machined with standard tooling"
```

```
:hasCtx Production ;This specifies that the above axioms only pertains to the Aero Engine Disc
Production Ontology which has the context 'Production'
```

```
(=> (and (Production.Rim ?rim)
  (MCCO.Form ?groove)
  (MCCO.hasAttributeOfInterest ?rim ?groove)
  (MCCO.NeckWidth ?nw)
  (MCCO.hasParameter ?groove ?nw)
  (MCCO.hasValue ?nw (MCCO.mm ?real1)))
```

```
(gteNum ?real1 10))
```

```
:IC hard "The NeckWidth of Rim should be greater than 10mm for the Rim to be machined with standard
tooling"
```

```
:hasCtx Production
```

```
(=> (and (Production.Rim ?rim)
```

```

(MCCO.Form ?groove)
(MCCO.hasAttributeOfInterest ?rim ?groove)
(MCCO.GrooveAngle ?ga)
(MCCO.hasParameter ?groove ?ga)
(MCCO.hasValue ?nw (MCCO.deg ?real1)))
(gteNum ?real1 45))
:IC hard "The GrooveAngle of Rim should be greater than 45deg for the Rim to be machined with standard
tooling"
:hasCtx Production

```

;=====Axioms to capture the predictive machining knowledge about the WebProfile =====

```

(=> (and (WebProfile ?wp)
          (MCCO.Form ?profile)
          (MCCO.hasAttributeOfInterest ?wp ?profile)
          (OuterRadius ?R)
          (InnerRadius ?r)
          (MCCO.hasParameter ?profile ?R)
          (MCCO.hasParameter ?profile ?r)
          (MCCO.hasValue ?R (MCCO.mm ?RValue))
          (MCCO.hasValue ?r (MCCO.mm ?rValue))
          (numPlus ?RValue ?rValue ?Rplusr))
      (hasRplusrValue ?wp (MCCO.mm ?Rplusr))))

```

```

;Rule to Infer WebProfile R-r
:Rel hasRminusrValue
:Inst BinaryRel
:Inst IntensionalRel
:Sig WebProfile MCCO.LinearDimension

```

```

(<= (hasRminusrValue ?wp (MCCO.mm ?Rminusr))
    (and (WebProfile ?wp)
          (MCCO.Form ?profile)
          (MCCO.hasAttributeOfInterest ?wp ?profile)
          (OuterRadius ?R)
          (InnerRadius ?r)
          (MCCO.hasParameter ?profile ?R)
          (MCCO.hasParameter ?profile ?r)
          (MCCO.hasValue ?R (MCCO.mm ?RValue))
          (MCCO.hasValue ?r (MCCO.mm ?rValue))
          (numMinus ?RValue ?rValue ?Rminusr))))

```

```

;To Infer (R-r)/0.15

```

```

:Rel hasRminusrbyfeed

```

```

:Inst BinaryRel
:Inst IntensionalRel
:Sig WebProfile MCCO.LinearDimension

(<= (hasRminusrbyfeed ?wp (MCCO.mm ?Rminusrbyfeed))
  (and (hasRminusrValue ?wp (MCCO.mm ?Rminusr))
    (numDivide ?Rminusr 0.15 ?Rminusrbyfeed)))

;Infer (R+r)/0.3183
:Rel hasRplusrbyconstant
:Inst BinaryRel
:Inst IntensionalRel
:Sig WebProfile MCCO.LinearDimension

(<= (hasRplusrbyconstant ?wp (MCCO.mm ?Rplusrbyconstant))
  (and (hasRplusrValue ?wp (MCCO.mm ?Rplusr))
    (numDivide ?Rplusr 0.3183 ?Rplusrbyconstant)))

; Rule to Infer SCL of WebProfile
:Rel WebProfilehasSCL
:Inst BinaryRel
:Inst IntensionalRel
:Sig WebProfile MCCO.LinearDimension

(<= (WebProfilehasSCL ?wp (MCCO.mm ?wpSCL))
  (and (WebProfile ?wp)
    (hasRplusrbyconstant ?wp (MCCO.mm ?Rplusrbyconstant))
    (hasRminusrbyfeed ?wp (MCCO.mm ?Rminusrbyfeed))
    (numMultiply ?Rplusrbyconstant ?Rminusrbyfeed ?wpSCL)))

;Axiom for Predicting machinability of WebProfile
(=> (and (WebProfile ?wp)
  (WebProfilehasSCL ?wp (MCCO.mm ?wpSCL)))
  (exists (?ct ?ctSCL)
    (and(MCCO.CuttingTool ?ct)
      (cuttingToolhasSCL ?ct (MCCO.mm ?ctSCL))
      (gteNum ?ctSCL ?wpSCL))))

:IC hard "The available tools cannot machine the asserted WebProfile in a single pass"

```