

**A Novel Virtual Product Modelling Framework for Design
Automation in a Knowledge-Based Engineering Environment**

Guolong Zhong

A thesis submitted to Birmingham City University in partial fulfilment

for the degree of

DOCTOR OF PHILOSOPHY

April 2022

Faculty of Computing, Engineering and Built Environment (CEBE)

Birmingham City University

Dedicated to

My parents Xinxiang Zhong and Juya Zhang

Abstract

Computer Aided Design (CAD) has been widely used for product modelling in the industry, where multiple issues arise, such as lack of product data representation and capturing and reusing the existing design knowledge in the modelling process. Existing CAD systems only provide geometric data within the CAD models and require users to have knowledge of the product to judge the correctness of the modelling process. Knowledge-Based Engineering (KBE) has been introduced to assist product design with the capabilities of knowledge capturing and reusing. However, there is always a “black box” problem in understanding the existing KBE applications, and the substantiation steps for the implementation of KBE frameworks are still limited. To address this, the author proposed and implemented a Virtual Product Modelling (VPM) framework that helps capture and reuse existing product information to enhance the modelling process for design automation. This framework was built as a knowledge-based product modelling environment using a gaming engine. It was further evaluated through three use cases, where the proposed framework was applied to simple parts with primitive geometric features, a hex bolt, and a wheel assembly. The results of the use case evaluation indicate that this framework satisfies all the identified measurement parameters and achieves the aim of the research. This research enhances the product modelling process with the capabilities of generative representation, knowledge capturing and reusing. It provides design engineers with the knowledge reasoning capability when they are making changes to the product model and, therefore, saves time and prevents engineers from making mistakes. This research also presents a KBE implementation framework with detailed substantiation steps, where the knowledge is structured and reusable within the product model. Further, the findings of this research have shown the potential of the developed VPM framework in aspects such as standard development in product modelling, extending to non-engineers and integration with VR/AR visualisation techniques.

Acknowledgements

In the beginning, I would like to say a huge thank you to my supervisors, Dr Venkatesh Vijay and Dr Noel Perera, for their tremendous support and guidance throughout my research journey. This PhD would not have been achievable without their patient encouragement and advice. In particular, my deepest gratitude goes to Vijay for motivating me and providing endless support during my whole PhD research.

Also, I would like to express my sincere thanks to all the people who were part of the research team at different times. That includes Professor Craig Chapman, Dr Raju Pathmeswaran and Professor Illisa Oraifige. Their precious support and comments provided me with a lot of helpful information and ideas to conduct this research and improve my work. I am very grateful to Professor Craig Chapman and Dr Raju Pathmeswaran for acknowledging my skills and offering me the opportunity to join the Knowledge-Based Engineering (KBE) Lab in the UK. It was a great honour to work with them. Identifying the scope of this research could not have been done without their vision and suggestions.

I am also thankful to Professor Peter Larkham and Sue Witton for their support in solving all the administrative issues during my PhD studies. My special thanks also go to Professor Lynsey Melville for her understanding and consistent support during my daily work. Without her support, it would not have been possible for me to focus on and complete my thesis.

I would like to thank my friends Maxim Filimonov, Sherdon Niño Uy, Nuo Lin at Birmingham City University for their continuous support and encouragement throughout each stage of my research. Discussions with them helped me greatly in my academic and scientific research growth. Finally, I would like to thank my parents for always supporting me and providing me encouragement and motivation to accomplish my PhD study.

Table of Contents

Abstract.....	I
Acknowledgements	II
Table of Contents	III
List of Figures.....	X
List of Tables	XVI
List of Publications	XVIII
List of Abbreviations	XX
1 Introduction and Background	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Research Questions	4
1.4 Research Aims and Objectives.....	5
1.5 Structure of the Thesis.....	6
1.6 Chapter Summary.....	7
2 Product Modelling and Design Engineering Automation	8
2.1 Introduction	8
2.2 Product Design for Product Development	8
2.2.1 Product Design: Advancement with Computer Aided Design (CAD)	11
2.2.2 From CAD to CAE	12
2.2.3 From CAD to CAM and CIM.....	12
2.2.4 From CAD to Computer Aided Product Modelling	13
2.2.5 Product Model.....	14

2.3	Design Engineering Automation.....	15
2.3.1	CAD Models (Geometric Models).....	15
2.3.2	Automation with CAD Models.....	17
2.3.3	Extension of CAD Models.....	19
2.3.4	Product Data Management and Product Lifecycle Management	20
2.4	Product Modelling Methods.....	21
2.5	Product Model Development	24
2.6	Chapter Summary.....	28
3	Knowledge-Based Engineering Techniques, Product Modelling Tools and Standards.....	30
3.1	Introduction	30
3.2	Knowledge-Based Engineering (KBE)	30
3.2.1	KBE and CAD	31
3.2.2	KBE Methods.....	31
3.3	KBE with Model-Based Engineering.....	38
3.3.1	Model-Based Engineering Overview.....	38
3.3.2	Applying Key Concepts from MBE and MBSE into KBE	39
3.4	Standards and Formats in Product Modelling.....	41
3.4.1	Product Modelling Formats and Standards.....	41
3.4.2	Modelling with STEP - EXPRESS and EXPRESS-G.....	44
3.4.3	STEP Application Protocols - AP203, AP214 and AP242.....	44
3.4.4	Relevant Research Work in Product Modelling with STEP	46

3.5	Tools in Product Modelling.....	48
3.5.1	Traditional CAD Software.....	49
3.5.2	Adaptive Modelling Language (AML).....	51
3.5.3	Gaming Engines in Product Modelling.....	51
3.6	Key Concepts of DEA with KBE System Development	53
3.6.1	Multi-Fidelity.....	53
3.6.2	Generative Modelling	55
3.6.3	Common Computational Model	55
3.6.4	Design Optimisation	56
3.6.5	Applying Key Concepts in KBE Product Modelling System Development	56
3.7	Related Research Work in KBE Product Modelling Framework Development.....	57
3.8	Literature Synthesis.....	59
3.8.1	Literature Review Summary	59
3.8.2	Research Gap Summary.....	62
3.8.3	Need for Knowledge Capture and Reuse in Product Modelling and Expected Contribution to Knowledge.....	63
3.9	Chapter Summary.....	65
4	Research Methodology	67
4.1	Introduction	67
4.2	Research Questions and Hypothesis Development.....	67
4.3	Research Plan	69
4.4	Research Phases	70

4.4.1	Phase1: Literature Review	71
4.4.2	Phase 2: Research Design (Enabling Methods).....	71
4.4.3	Phase 3: Development.....	79
4.4.4	Phase 4: Evaluation with Use Cases	80
4.5	Chapter Summary.....	83
5	Virtual Product Modelling Framework.....	84
5.1	Virtual Product Modelling Framework Development	84
5.1.1	Product Model Development	87
5.1.2	Knowledge Capture of Non-Geometric Information.....	88
5.1.3	Knowledge Capture of Geometric Information	90
5.1.4	Knowledge Mapping.....	91
5.1.5	Product Visualisation and Validation	93
5.2	Product Model Development in VPM.....	93
5.2.1	Meta Class of VPM.....	94
5.2.2	Description of VPM Knowledge Classes	97
5.2.3	VPM Data Exchange Method and Format.....	99
5.3	Overall Virtual Product Modelling Framework Implementation Methods.....	100
5.3.1	Knowledge Source	100
5.3.2	Knowledge Capture	101
5.3.3	Knowledge Store and Exchange	102
5.3.4	Knowledge Mapping.....	103
5.3.5	Visualisation	104

5.4	Chapter Summary.....	106
6	Evaluation.....	108
6.1	Introduction	108
6.2	Evaluation Objectives	109
6.3	Use Case 1: Simple Part with Primitive Design Features	113
6.3.1	Use Case Overview	113
6.3.2	Simple Part – a Block Part	113
6.3.3	Simple Part – a Cylinder Part.....	128
6.3.4	Simple Part – a Cone Part	136
6.3.5	Simple Part - a Sphere Part	145
6.3.6	Result Analysis and Use Case Discussion	153
6.4	Use Case 2: Basic Engineer Part – Hex Bolt	161
6.4.1	Use Case Overview	161
6.4.2	Measurement Parameters and Testing Scenarios.....	164
6.4.3	Virtual Product Modelling Framework Application.....	166
6.4.4	Result Analysis and Use Case Discussion	179
6.5	Use Case 3: Engineer Assembly – Wheel Assembly	187
6.5.1	Use Case Overview	187
6.5.2	Measurement Parameters and Testing Scenarios.....	193
6.5.3	Virtual Product Modelling Framework Application.....	195
6.5.4	Result Analysis and Use Case Discussion	216
6.6	Discussion and Findings.....	223

6.7	Chapter Summary.....	230
7	Conclusion and Recommendation	232
7.1	Introduction	232
7.2	Summary	232
7.3	Research Outcomes	240
7.3.1	Research Question 1	240
7.3.2	Research Question 2	241
7.3.3	Research Question 3	242
7.3.4	Research Question 4	243
7.3.5	Research Hypothesis 1	244
7.3.6	Research Hypothesis 2	244
7.3.7	Conclusions.....	245
7.4	Contribution to Knowledge.....	247
7.4.1	Virtual Product Modelling Framework.....	247
7.4.2	Virtual Product Modelling Structure	247
7.4.3	Knowledge Capture and Data Exchange Method.....	248
7.4.4	Substantiation for KBE System Implementation.....	249
7.5	Limitations of Research	249
7.6	Recommendations for Future Work.....	251
7.6.1	Improving the Knowledge-Based Product Modelling Tool.....	251
7.6.2	Enhancing Data Exchange in Product Modelling with New Product Modelling Standard AP242	252

7.6.3	Extending the Product Modelling Standards for Knowledge Capture and Reuse Implementation	252
7.6.4	Implementing the Framework for Non-Engineers	253
7.6.5	KBE Application Development Using a Gaming Engine.....	253
7.6.6	Product Modelling with Virtual Reality and Augmented Reality Technology	254
Reference	255
Appendix 1: Knowledge Capture Tool Interface Maximised View	270
Appendix 2: Scripts Used in Knowledge Mapping in Use Case 1	272
Appendix 3: Scripts Used in Knowledge Mapping in Use Case 2	280
Appendix 4: Scripts Used in Knowledge Mapping in Use Case 3	298
Appendix 5: Scripts Used in Parsing Data From Knowledge File	314
Appendix 6: Full-Text Papers of Publications	328

List of Figures

Figure 2-1: Four stages of the design process. Adapted from Gerhard Pahl and Wolfgang Beitz (1988).	9
Figure 2-2: Manpower requirements in the evolution of an engineering design problem. Adapted from Ross and Ward (1968).	11
Figure 2-3: The Evolution of Product Development. Adapted from Krause <i>et al.</i> (1993).	13
Figure 2-4: Development of PDM. Source: Moorthy and Vivekanand (2007)	21
Figure 2-5: Possible content of the Product Model. Adapted from Isaksson <i>et al</i> (2000).	25
Figure 3-1: KBE application lifecycle from MOKA. Source: Melody Stokes (2001).	32
Figure 3-2: The KNOPRESSA modelling set. Source: Lovett, Ingram and Bancroft (2000). 33	
Figure 3-3: DEE process flow to support multidisciplinary design optimisation. Source: Berends, Van Tooren and Schut (2008).....	34
Figure 3-4: The KNOMAD Methodology. Source: Curran, Verhagen and Van Tooren (2010).	35
Figure 3-5: The eleven KCM steps. Adapted from Terpenney, Strong and Wang (2000).	36
Figure 3-6: Converting HFM to LFM by different simplifications. Adapted from Fernández-Godino et al. (2016).	54
Figure 4-1: Research plan	70
Figure 4-2: Implementation framework for the development.....	80
Figure 5-1: Virtual product modelling framework	85
Figure 5-2: Developed product model structure from VPM.....	88
Figure 5-3: Task flow of capturing the non-geometric information	89
Figure 5-4: Task flow of capturing the geometric information	91

Figure 5-5: Knowledge mapping framework for knowledge reasoning	92
Figure 5-6: Example of the relationship between the assembly and parts of the product model (named “Product A”)	94
Figure 5-7: Product model structure data from literature. Adapted from Fenves <i>et al.</i> , (2004), Siemens PLM (2019b) and Boy <i>et al.</i> , (2015).	95
Figure 5-8: Description of VPM data exchange method	99
Figure 5-9: The developed knowledge capture tool interface (maximised view of this tool interface is provided in Appendix)	102
Figure 5-10: Example of a simplified XML schema for the Knowledge File	103
Figure 5-11: Example of knowledge mapping logic for implementation	104
Figure 5-12: Example of visualisation in the developed knowledge-based product modelling environment	106
Figure 6-1: Four simple parts with primitive design features (modelled in Siemens NX 10)	113
Figure 6-2: VPM product model structure of the block part in UML diagram	116
Figure 6-3: Example - select the number of the parts being designed	117
Figure 6-4: Example - input the Product Information including name, description, and type	117
Figure 6-5: Example - input the Design Intent and its description	118
Figure 6-6: Example - input the Function	118
Figure 6-7: Example - input the Form	118
Figure 6-8: Example - input the Material	119
Figure 6-9: Example - input the Behaviour	119

Figure 6-10: Example - input the Rules.....	120
Figure 6-11: Example - input the Fit.....	120
Figure 6-12: Example - input the Relationship.....	121
Figure 6-13: Example - input the Dimension	121
Figure 6-14: Example - input the Key Parameters.....	122
Figure 6-15: Example - input the Constraints.....	122
Figure 6-16: Example - part of knowledge file of the block part	123
Figure 6-17: VPM product model structure of the block part with captured knowledge	124
Figure 6-18: A block part model created in Siemens NX 10.....	125
Figure 6-19: Part of the step file of block that exported from Siemens NX 10.....	125
Figure 6-20: Illustration of the knowledge mapping for the block part in use case 1	126
Figure 6-21: Block part model visualised in the developed knowledge-based product modelling environment.....	127
Figure 6-22: Results of validation – use case 1: Block part	128
Figure 6-23: VPM product model structure of the cylinder part in UML diagram	130
Figure 6-24: Example - part of knowledge file of the cylinder part	131
Figure 6-25: VPM product model structure of the cylinder part with captured knowledge..	132
Figure 6-26: A cylinder model created in Siemens NX 10.....	132
Figure 6-27: Illustration of the knowledge mapping for the cylinder part in use case 1	133
Figure 6-28: Cylinder part model visualised in the developed knowledge-based product modelling environment.....	134
Figure 6-29: Results of validation – use case 1: cylinder part.....	135
Figure 6-30: VPM product model structure of the cone part in UML diagram.....	138

Figure 6-31: Example - part of knowledge file of the cone part.....	139
Figure 6-32: VPM product model structure of the cone part with captured knowledge	140
Figure 6-33: A cone model created in Siemens NX 10	141
Figure 6-34: Illustration of the knowledge mapping for the cone part in use case 1.....	142
Figure 6-35: Cone part model visualised in the developed knowledge-based product modelling environment.....	143
Figure 6-36: Results of validation – use case 1: cone part	145
Figure 6-37: VPM product model structure of the sphere part in UML diagram.....	147
Figure 6-38: Example - part of knowledge file of the sphere part.....	148
Figure 6-39: VPM product model structure of the sphere part with captured knowledge	149
Figure 6-40: A sphere model created in Siemens NX 10	150
Figure 6-41: Illustration of the knowledge mapping for the sphere part in use case 1.....	151
Figure 6-42: Sphere part model visualised in the developed knowledge-based product modelling environment.....	152
Figure 6-43: Results of validation – use case 1: sphere part.....	153
Figure 6-44: A hex bolt modelled in Siemens NX 10	161
Figure 6-45: 2D drawings of the hex bolt with dimensional descriptions.....	162
Figure 6-46: VPM product model structure of the hex bolt in UML diagram	166
Figure 6-47: Example - part of knowledge file of the hex bolt	167
Figure 6-48: VPM product model structure of the hex bolt with captured knowledge	168
Figure 6-49: Illustration of the knowledge mapping for the scenario one – changing L under one D1 in use case 2.....	170

Figure 6-50: Illustration of the knowledge mapping for the scenario one – changing b under one D1 in use case 2.....	171
Figure 6-51: Illustration of the knowledge mapping for the scenario one – changing b under one D1 in use case 2.....	172
Figure 6-52: Illustration of the knowledge mapping for the scenario two – changing D1 in use case 2.....	173
Figure 6-53: Hex bolt model visualised in the developed knowledge-based product modelling environment	174
Figure 6-54: Results of validation – use case 2: hex bolt, scenario one - change L	175
Figure 6-55: Results of validation – use case 2: hex bolt, scenario one - change b	176
Figure 6-56: Results of validation – use case 2: hex bolt, scenario one - change k	177
Figure 6-57: Results of validation – use case 2: hex bolt, scenario one - change e.....	177
Figure 6-58: Results of validation – use case 2: hex bolt, scenario one - change s.....	178
Figure 6-59: Results of validation – use case 2: hex bolt, scenario two – change bolt thread size D1 from M12 to M14	179
Figure 6-60: A wheel assembly modelled in Siemens NX 10.....	187
Figure 6-61: 2D drawings of the wheel part with dimensional descriptions	188
Figure 6-62: 2D drawings of the tyre part with dimensional descriptions	189
Figure 6-63: Virtual product model structure of the wheel assembly in UML diagram	198
Figure 6-64: Examples - parts of knowledge files of the wheel assembly, wheel part and tyre part	200
Figure 6-65: Virtual product models of the wheel assembly, wheel part and tyre part in UML diagram	203
Figure 6-66: Wheel part modelled in Siemens NX 10.....	204

Figure 6-67: Tyre part modelled in Siemens NX 10.....	205
Figure 6-68: Illustration of the knowledge mapping for the wheel part rules	206
Figure 6-69: Illustration of the knowledge mapping for the tyre part rules.....	207
Figure 6-70: Illustration of the knowledge mapping for the wheel assembly rules.....	209
Figure 6-71: Illustration of the knowledge mapping for the testing scenario one and two in use case 3	210
Figure 6-72: Wheel assembly model visualised in the developed knowledge-based product modelling environment	211
Figure 6-73: Functions of making possible changes to the wheel assembly model in the developed knowledge-based product modelling environment	212
Figure 6-74: Results of validation – use case 3: wheel assembly, scenario one - change the wheel part dimension - L1 parameter	214
Figure 6-75: Results of validation – use case 3: wheel assembly, scenario two – change of the tyre part dimension – S2 parameter	216

List of Tables

Table 2-1: Existing product model development work to support generative representation .	26
Table 3-1: Focus areas of reviewed KBE methodologies.....	37
Table 3-2: Comparison between IGES, STEP and JT	43
Table 3-3: Reviewed traditional CAD software (version up to 2021) in this research.....	50
Table 3-4: Characteristics of a product modelling system from the KBE perspective.....	57
Table 4-1: Comparison of product modelling methods	73
Table 4-2: Key elements in related research work from literature	74
Table 4-3: Comparison between IGES, STEP and JT	75
Table 4-4: Reviewed implementation tools (version up to 2021) in this research	77
Table 4-5: Summary of solutions for solving the identified problems	78
Table 4-6: Design Evaluation Methods. Adapted from Hevner <i>et al.</i> (2004).....	80
Table 5-1: Mapping between KCM steps and the VPM framework	87
Table 5-2: Illustration of implementation method of rules in this research.....	92
Table 5-3: External geometric classes mapping with STEP Entity. Adapted from (Siemens PLM, 2019b).....	96
Table 5-4: Explanation of VPM knowledge classes.....	98
Table 6-1: Measurement parameters mapped with evaluation criteria	110
Table 6-2: Existing information of use case 1- primitive design feature: block part	114
Table 6-3: Measurement parameters and expected results of use case 1.....	115
Table 6-4: Existing information of use case 1- primitive design feature: cylinder part	129
Table 6-5: Existing information of use case 1- primitive design feature: cone part.....	136
Table 6-6: Existing information of use case 1- primitive design feature: sphere part.....	146

Table 6-7: Comparison of the use case 1 implementation results between the existing/legacy CAD system and VPM.....	158
Table 6-8: Hex bolt dimensions (in millimetres) – DIN 931 (partial).....	162
Table 6-9: Existing information of use case 2 – hex bolt	163
Table 6-10: Measurement parameters and expected results of use case 2- hex bolt	164
Table 6-11: Comparison of the use case 2 implementation results between the existing/legacy CAD system and VPM.....	183
Table 6-12: Wheel assembly parameters and rules.....	189
Table 6-13: Existing information of use case 3 – wheel assembly	191
Table 6-14: Existing information of use case 3 – wheel.....	192
Table 6-15: Existing information of use case 3 – tyre	193
Table 6-16: Measurement parameters and expected results of use case 3 – wheel assembly	194
Table 6-17: Comparison of the use case 3 implementation results between the existing/legacy CAD system and VPM.....	220

List of Publications

Paper:

Zhong, G., Vijay, V. C. and Oraifige, I. (2021) ‘A Game-Based Product Modelling Environment for Non-Engineer’, in *ICSGGBL 2021: 23rd International Conference on Serious Games and Game-Based Learning*.

Abstract:

Knowledge-Based Engineering (KBE) has shown its advantages in the last two decades in product development in different engineering areas such as automation, mechanical, civil and aerospace engineering in terms of digital design automation and cost reduction. However, in the primary design stages, the descriptive information of a product is discrete and unorganized, while knowledge is in various forms instead of pure data. Thus, it is crucial to have an integrated product model which can represent the entire product information and its associated knowledge at the beginning of the product design. One of the shortcomings of the existing product models is a lack of required knowledge representation in various aspects of product design and its mapping to an interoperable schema. This paper introduced a method to provide a general product model as a generative representation of a product, which consists of the geometric information and non-geometric information, through a product modelling framework. The proposed method for capturing the knowledge from the designers through a knowledge file provides a simple and efficient way of collecting and transferring knowledge. Further, the knowledge schema provides a clear view and format of the data that needed to be gathered to achieve a unified knowledge exchange between different platforms. This study used a game-based platform to make the product modelling environment accessible for non-engineers. Further, the paper goes on to test the use case based on the proposed game-based product modelling environment to validate the effectiveness among non-engineers.

Paper (under review):

Zhong, G., Vijay, V. C. and Perera, N. (2022) ‘Enhancing Engineering Product Design using a Knowledge Based Game Engine Platform’, *Advances in Engineering Software*.

Abstract:

Traditional CAD tools and systems cannot explain the real-world concepts by themselves and require the users to have knowledge and design experience of the product in order to understand design rules and judge the correctness of the changes. Knowledge-Based engineering (KBE) has been introduced to address these issues; however, existing KBE methodologies offer limited instantiation steps and enabling tools for implementation. In this paper, a knowledge-based product modelling prototype system is proposed to overcome the above problems. This system is developed based on the Virtual Product Modelling framework using a game engine platform to aid in capturing, reusing, and exchanging the existing product information and provide knowledge reasoning in the product modelling process. Three different use cases were applied in the research to validate the effectiveness of the proposed system. The use case evaluation has proved that the developed prototype system can help save time and prevent engineers from making mistakes in the product design. The findings of this research have shown the potential of integrating product modelling with VR/AR visualisation techniques and applying this system to non-engineers.

List of Abbreviations

3D = 3 Dimensional

AISI = American Iron and Steel Institute

AML = Adaptive Modelling Language

ASME = American Society of Mechanical Engineers

AP = Application Protocol

API = Application Programming Interface

AR = Augmented Reality

BOM = Bills of Material

B-rep = Boundary Representation

CAD = Computer Aided Design

CAE = Computer Aided Engineering

CAM = Computer Aided Manufacturing

CCM = Common Computational Model

CFD = Computational Fluid Dynamics

CIM = Computer Integrated Manufacturing

CPM = Core Product Model

CSG = Constructive Solid Geometry

DEA = Design Engineering Automation

DEE = Design and Engineering Engine

DIN = Deutsches Institut für Normung (German Institute for Standardisation)

EDO = Engineering Design Optimisation

ERP = Enterprise Resource Planning

FEA = Finite Element Analysis

GUI = Graphical User Interface

HFM = High-Fidelity Model

HTML = HyperText Markup Language

ICARE forms = Illustration, Constraint, Activity, Rules, and Entity forms

IGES = The Initial Graphics Interchange Specification

INCOSE = The International Council on Systems Engineering

ISO = International Organisation for Standardisation

JT = Jupiter Tessellation

KBE = Knowledge-Based Engineering

KCM = Knowledge Capture Methodology

KF = Knowledge File

KIC = Knowledge-Intensive CAD

KNOMAD = Knowledge Nurture for Optimal Multidisciplinary Analysis and Design

KOMPRESSA = Knowledge Oriented Methodology for the Planning and Rapid Engineering
of Small-Scale Application

LFM = Low-Fidelity Model

MBE = Model-Based Engineering

MBSE = Model-Based Systems Engineering

MOKA = Methodology and tools Oriented to Knowledge-based engineering Application

OMG = Object Management Group

OWL = Ontology Web Language

PACKS = The Parametric Composite Knowledge System

PDM = Product Data Management

PHP = Hypertext Preprocessor

PLM = Product Lifecycle Management

PMI = Product and Manufacturing Information

SCAD = The Steered Composite Analysis and Design System

SMEs = Small to Medium Enterprises

STEP = Standard for the Exchange of Product model data

SysML = System Modelling Language

UML = Unified Modelling Language

XML = Extensible Markup Language

VPM = Virtual Product Modelling

VR = Virtual Reality

1 Introduction and Background

1.1 Introduction

The rapid development of science and information technologies has generated higher demands for industrial development capacity, productivity, and agile response to the market. They all drive the industry to design and produce more complex products at lower costs and with less time. Computer Aided Design (CAD) has been introduced as a Design Engineering Automation (DEA) method for completing product design. Nonetheless, CAD tools and systems cannot understand and explain real-world design concepts by themselves. To judge the correctness of the design, CAD tools and systems require users to have sufficient knowledge and design experience of the product. However, the knowledge of a product is often discrete, unorganised and in various forms (Suresh and Egbu, 2006). There is a lack of required knowledge representation in multiple aspects of product design.

Product modelling has been regarded as a pivotal role to develop product models. Successful product modelling can support and improve the product development process chains throughout the product life cycle, resulting in cost reduction and time optimisation. A product model representing all detailed design information will enable designers to work on the design tasks without previous design experience. Time and cost on tutorial sessions and training for new designers could be saved. Existing research works show that there are two challenges in integrating extra data with CAD models: capturing and managing the product data in the complex product models using traditional CAD (Cooper, Fan and Li, 2001; Chang, 2015) and the mismatch between the availability of information and the accessibility of the appropriate information to designers (Blessing and Wallace, 1998). Thus, there is a need to develop a product model which can capture and reuse complex product data and provide accessible and appropriate information to designers.

In the last 20 years, Knowledge-Based Engineering (KBE) has shown its advantages in product development in different engineering areas such as automation, mechanical engineering, civil engineering and aerospace engineering in terms of modelling and cost reduction. It helps automate the repetitive design tasks by capturing, integrating, utilising and reusing existing knowledge required in various aspects of the product design (Chapman *et al.*, 2007; Rocca, 2012). The use of KBE methods and techniques has played an important role in design engineering automation for the development of a product in the industry (Shehab and Abdalla, 2001; Sanya and Shehab, 2014). The trend of product modelling for design engineering automation has evolved from manual drafting to CAD, from CAD to Computer Aided Product Modelling, and then to Knowledge-Based Product Modelling. The product data involved in product modelling has been expanded from “geometry-only” to “knowledge-integrated”. However, existing product models and CAD tools show limited capabilities in capturing and reusing knowledge. Further, the substantiation steps for implementing the current KBE methods and techniques for product modelling are usually not available and understandable to users (Cederfeldt, Elgh and Rask, 2006; Fan and Bermell-Garcia, 2008). Therefore, to overcome this “black-box” problem, it is necessary to develop a KBE implementation framework along with use cases and enabling tools for the purpose of capturing and reusing design knowledge in product modelling for design engineering automation.

1.2 Problem Statement

In a traditional product modelling environment, the designer can create and provide the visible geometry information in the product model. However, only geometry information is not enough to describe a product model completely. The utilisation of the knowledge can significantly reduce the unnecessary re-analysis, re-design, and re-planning, simplify the modelling tasks and ensure the modelling quality. A generative and efficient representation of

a product requires detailed geometry information for visualisation and the associated knowledge, such as function, form, behaviour, design rules, material, etc.

Capturing and transferring the knowledge of experienced designers are difficult. According to Suresh and Egbu (2006), the main challenge existing in small and medium companies and industries for implementing knowledge capture initiatives is a lack of awareness of knowledge capture benefits. Due to this, individuals and small and medium enterprises are lacking the vision and strategy and structure for knowledge capture. They have a strong reliance on informal networks and collaboration to locate the repository of knowledge.

Therefore, a critical issue of knowledge-based product modelling is capturing, classifying, structuring, and managing the captured knowledge. Since there is no clear formalised link between a generic product model and an interoperable format in the KBE environment, it is essential to provide well-defined knowledge classes and a formalised knowledge capture method for individuals, enterprises and industries to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats. In this thesis, geometric data contained in a CAD file is regarded as “geometry information”, and non-geometric information, such as experience, expertise and design rules, is called “knowledge”.

This research aims to provide a product modelling framework and tool that could capture, reuse and integrate the associated knowledge applied by designers into a generative product model. To achieve this goal, three key factors were considered. Firstly, the generated product model must have well-defined classes that can represent the entire product information and its associated knowledge. Then, the product model needs to be represented in an interoperable schema to ensure a steady data exchange between different product modelling platforms and CAD software. Third, the captured knowledge could be reused to enhance the product modelling process.

In this research, the author proposed and implemented a Virtual Product Modelling (VPM) framework that help in capturing and reusing existing product information to enhance the product modelling process for design automation. In contrast to the existing/legacy CAD systems, the proposed VPM framework has shown its advantages in generative representation, capturing, reusing and visualising the existing product knowledge to enhance the product modelling process. It provides design engineers with capabilities of knowledge reasoning when they are making changes to the product model to prevent them from making mistakes in the modelling process. The geometric data and the existing product knowledge are captured and integrated as one holistic product model through the use of VPM. Both the geometric data and the knowledge are stored and transferred in interoperable and neutral formats. This ensures that the captured knowledge can be exchanged along with the geometric data in one product model between different platforms. Compared to the CAD model generated from existing/legacy CAD systems, the design knowledge is structured and reusable within the developed product model using VPM. This provides a clear representation of the complex product knowledge and helps users understand the product and identify and access the essential product information.

The rest of the thesis will discuss in detail how the design engineer's knowledge is captured, reused, and mapped into the framework to enhance the product modelling process for design automation.

The following are the research questions and objectives to address the research challenges mentioned above.

1.3 Research Questions

- How can the design knowledge be structured and represented through a product model?

- How can this product model be implemented in a knowledge-based product modelling environment?
- How can the principles and practice of knowledge-based engineering be applied to capture and reuse the existing design knowledge for product modelling through a knowledge-based product modelling framework?
- How can this framework be implemented and applied by designers to enhance product modelling?

1.4 Research Aims and Objectives

This research aims to identify and develop methods and tools for capturing, reusing and exchanging existing design knowledge to enhance the product modelling process for design automation in a knowledge-based product modelling environment. The aim will be accomplished through the successful achievement of the following research objectives:

- To establish the research scope by identifying and reviewing the features and issues on product modelling for design engineering automation.
- To review methods, standards and tools used in product modelling for developing product models.
- To distinguish appropriate enabling methods and tools for capturing and reusing knowledge in product modelling.
- To develop methods of capturing, reusing, and exchanging design knowledge for product modelling and to develop a knowledge-based product modelling environment for applying these methods.
- To validate proposed methods of capturing, reusing and exchanging design knowledge in product modelling and evaluate the performance of the developed knowledge-based product modelling environment.

- To evaluate the modelling process within the developed knowledge-based product modelling environment, in contrast to the modelling process in existing CAD systems.
- To generate implementation guidance on how to use the approach for capturing and reusing existing product design knowledge to support design automation.

1.5 Structure of the Thesis

The structure of the thesis is described in this section. This thesis consists of seven chapters. Chapter 1 of the thesis provides a brief introduction of the research background and states the problems in the research field. This chapter also presents the research questions addressed by the current work and the research objectives.

Chapter 2 of the thesis focuses on the literature review of the current methods in product modelling and design engineering automation. It discussed the evolving process of the product design for product development and its advancement with Computer Aided Design (CAD) technology. The deployment and development of product models for design engineering automation are also presented. Different product modelling and product models are discussed and compared in this chapter.

Chapter 3 of the thesis extends the literature review to current knowledge-based engineering techniques and product modelling tools and standards. This chapter discussed key concepts and various methods for capturing and reusing knowledge in the product modelling process and compared different potential product modelling tools and standards for implementation. It also reviewed relevant research work in product modelling with STEP standards and knowledge-based product modelling framework development. In the end, findings from the literature review are summarised, and research gaps are identified. In conclusion, the need for knowledge capture and reuse in product modelling and expected contribution to knowledge is presented.

Chapter 4 presented a research plan to explain the research through four phases: literature review, research design, development, and evaluation. This chapter also provides the identified enabling methods for undertaking research development within the realm of capturing and reusing existing product knowledge in product modelling of engineering components. It explained the applied methods in this research with justifications and explanations of why the selected methods best fit this research.

Chapter 5 of the thesis presents the virtual product modelling framework, which addresses the research gaps identified in the previous chapters. It shows the development process of this virtual product modelling framework and explains how it can be further implemented with these identified enabling methods.

Chapter 6 shows the evaluation of the framework through the implementation of use cases. Three testing use cases are selected from the literature to validate and evaluate the effectiveness and efficiency of the proposed methodology. Detailed instantiation steps of this virtual product modelling framework for each use case are presented. Discussions and findings from the use case implementation and evaluation are also provided in this chapter.

Chapter 7 summarises the research outcome by aligning with research questions and hypotheses based on the results. Further, it discusses the limitations of the research and concludes the thesis. Recommendations for future work are presented at the end.

1.6 Chapter Summary

This chapter provides a brief introduction to this research area and explores the research issues and challenges. It also outlines the research questions, aims and objectives for conducting the study. The chapter concludes with the thesis structure showing how the chapters were written and connected. Literature that related to key concepts and issues of the research is discussed in the next chapter.

2 Product Modelling and Design Engineering Automation

2.1 Introduction

This research aims to develop a product modelling approach for capturing and reusing the existing product knowledge to enhance the product modelling process for Design Engineering Automation (DEA). Therefore, the area of this research is mainly focused on four fields: Product Design, Design Engineering Automation, Product Modelling, and Knowledge-Based Engineering. This chapter begins by providing an overview of various aspects of product design to show a thorough understanding of product development. This chapter then presents how the CAD models and existing knowledge are reused for design engineering automation to state of the art. This identifies that there is a lack of required knowledge representation in current CAD models and the reuse of the existing knowledge in product modelling is limited. Next, different product modelling methods have been analysed and discussed in this chapter. In the end, as the development of aimed product modelling approach requires sufficient and accurate representation of the associated knowledge involved in the product design process, Knowledge-Based Engineering (KBE) has been identified as an appropriate method to help capture, integrate, utilise and reuse the existing knowledge required in various aspects of the product design. This will lead to a further in-depth discussion on knowledge-based engineering techniques in Chapter 3.

2.2 Product Design for Product Development

Product design concerns solving design problems from the initial design idea to the final product. It is defined as

“The activity in which ideas and needs are given physical form, initially as solution concepts and then as a specific configuration or arrangement of elements, materials and components”

(Walsh, Roy and Bruce, 1988)

The design process of a product is mainly defined into four stages (Gerhard Pahl and Wolfgang Beitz, 1988): (i) planning and clarification of the task, (ii) conceptual design, (iii) embodiment design, (iv) detail design (see Figure 2-1). However, the boundary between each stage of the design process always overlaps because of the iterative nature of the design process.

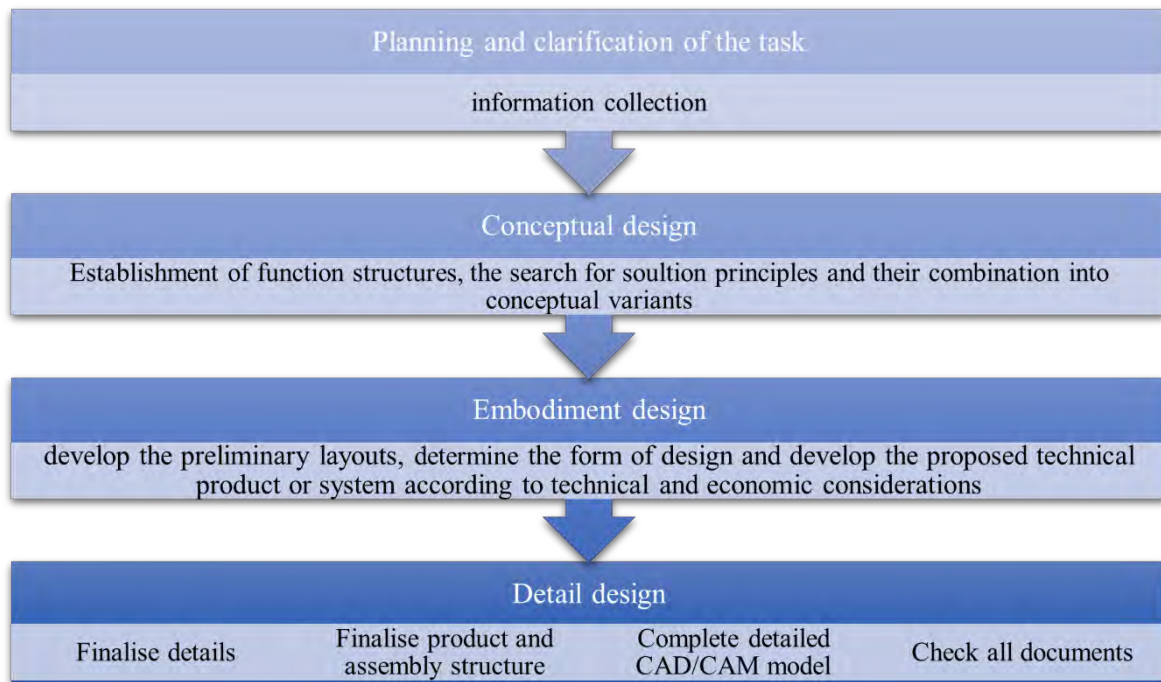


Figure 2-1: Four stages of the design process. Adapted from Gerhard Pahl and Wolfgang Beitz (1988).

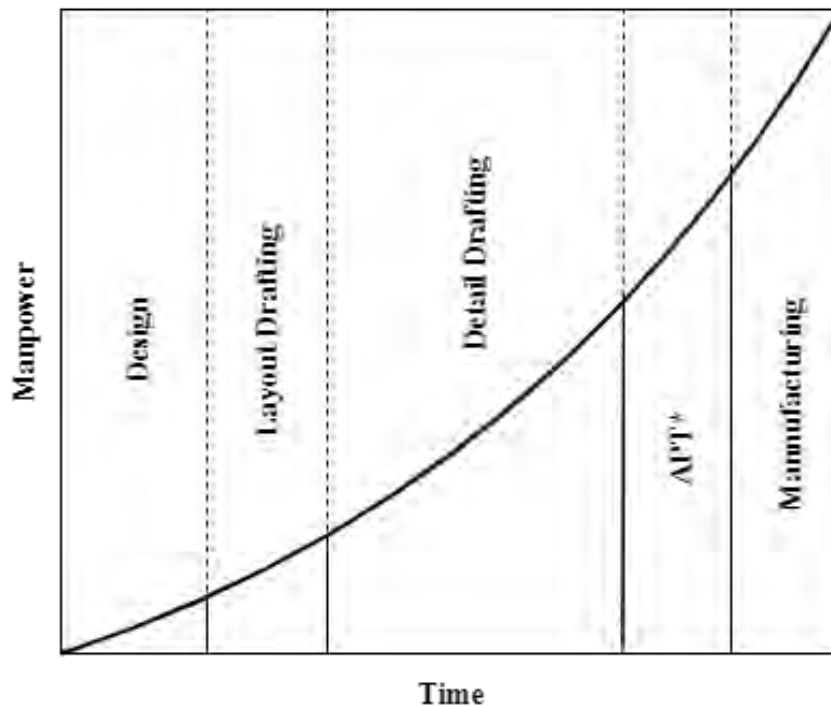
Stage 1 involves the planning of the initial product by collecting information about customer requirements. This stage results in initial product ideas and a detailed design specification list. The concept design stage involves the establishment of function structures, the search for solution principles and their combination into conceptual variants (Pahl *et al.*, 2007). Since the design specifications may range from a high-level abstract statement to detailed documentation, the analysis of functional requirements or product function is very crucial at this stage (Wang *et al.*, 2002). Designers need to be critical in their decision making because

once all concepts have been determined, it is difficult to correct fundamental problems at the later stages.

The concept design stage is followed by the embodiment design. In this stage, designers start to develop the preliminary product architecture, determine the shape and general dimensions or sizes of the design and develop the proposed technical product or system according to technical and economic considerations. By the end of the embodiment stage, key design parameters that could be controlled by the designer will be identified, for example, dimension, tolerance, material, etc. (Pahl *et al.*, 2007; Langeveld, 2011)

Detail design is the last stage in the product design process. Designers finalise design details, such as geometry, material, tolerances of all the parts, and the structure of the product and assembly in this stage (Pahl *et al.*, 2007; Johnson and Gibson, 2014). Complete detailed drafting and CAD/CAM models will be produced, which contain product shapes, forms, dimensions and surface properties, etc.

Coons and Mann (1960) pointed out that the detail or element design would be the area that is most conducive to machine assistance. Figure 2.2 shows the relationship between elapsed time and manpower in the evolution of an engineering design problem of Massachusetts Institute of Technology (MIT)'s Computer Aided Design project (Ross and Ward, 1968). Proceeding from left to right in the time sequence, it can be seen from the figure that as the design stage moves further, more and more manpower becomes involved in the project. According to the design process defined by Gerhard Pahl and Wolfgang Beitz (1988), shown in Figure 2-1, detail or element design is covered in the embodiment stage and detail design stage. Thus, the most time-consuming parts of the design process are the embodiment design and detail design (Langeveld, 2011).



Note: *Automatically Programmed Tools (APT) is used to program numerically-controlled machine tools to create complex parts using a cutting tool moving in space.

Figure 2-2: Manpower requirements in the evolution of an engineering design problem.

Adapted from Ross and Ward (1968).

2.2.1 Product Design: Advancement with Computer Aided Design (CAD)

To reduce the elapsed time, manpower and resources expended in completing the design process, computer systems have been introduced to ease the design of parts and tools for the industry since the early 1960s (Carlson, 2017). The use of computers in creating, modifying, analysing, or optimising a design was a revolutionary step in product design. The current CAD tools and systems provide designers with a wide variety of functionalities, such as creating 2D drafting and engineering drawings, building a 3D model of a product, visualisation of the product's 3D geometry shape, and the creating of assembly with product components (Weisberg, 2008). CAD tools and systems also allow designers to create Bill of Materials and engineering drawings that lay out all the required information of parts and assembly for the further manufacturing process.

2.2.2 From CAD to CAE

Computer Aided Engineering (CAE) is a broad concept that means using computers in all phases of engineering design work (Carlson, 2017). It involves not only the CAD but also engineering analysis tasks, for example, finite element analysis (FEA), computational fluid dynamics (CFD), product optimisation, etc. The key capability of CAE is in testing designs by simulating real-world conditions. Thus, CAE tools have been developed and used to test and analyse the performance of the designed product. However, a common CAD model is always required and used as a core data repository and supply source of input for CAE tools (G P Gujarathi and Ma, 2011). Simulation and analysis with CAD models have become a crucial process in current high-tech industries such as aerospace and semiconductors.

2.2.3 From CAD to CAM and CIM

When the design goes to the manufacturing stage, CAD will always be linked to Computer Aided Manufacturing (CAM). CAM is a subsequent process after CAD or CAE in most computer-aided product development. It allows users to control, monitor, and adjust machine tools in manufacturing workpieces (Carlson, 2017). CAM tools evolved from numerical controlled (NC) programming tools, which could generate the tool path and code for CNC machines and cutting tool parameters for machine operations. Modern CAM tools and systems can automatically generate tool paths from a 3D CAD model and simulate the cutting action.

With the increasing use of computers in design and manufacturing under CAD and CAM, Computer Integrated Manufacturing (CIM) became an acronym for the area where tasks are common to both CAD and CAM. CIM integrates the component data (geometry, tolerance, etc.) that was created with CAD into a database so that it can be reused for the CAM environment directly (Alavudeen and Venkateshwaran, 2008). Computerised information from CAD plays a significant role in CIM. Bozdoc (2003) stressed that a highly developed

CIM system required the creation of a database where all the information for manufacturing is stored in a form that can be retrieved and reused by anyone who needs it.

2.2.4 From CAD to Computer Aided Product Modelling

CAD tools and systems have shown great advantages in the creation, modification, analysis or optimisation of a product. However, CAD tools and systems cannot understand real-world concepts, such as the purpose of the product being designed or the function that the product will serve (Salzman, 1989). CAD tools and systems require the designer to have knowledge and design experience of the product to judge the correctness of the function and to understand what is going on beyond that which is graphically shown on the computer screen. Hence, there is a need to integrate product information such as the designer's idea into a formal product model through the design process while using CAD.

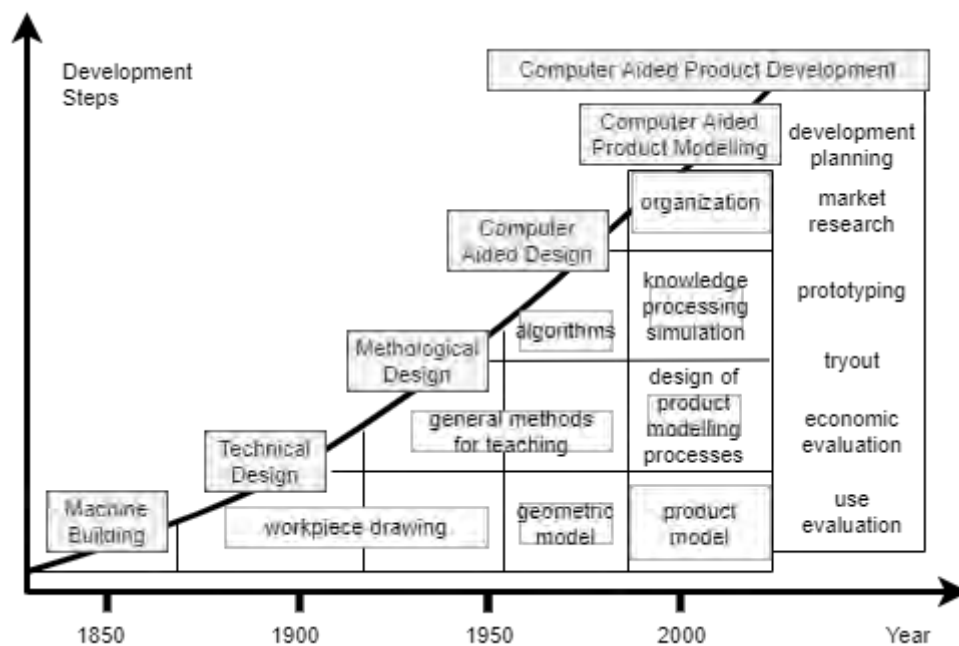


Figure 2-3: The Evolution of Product Development. Adapted from Krause *et al.*(1993).

The concept of computer aided product modelling was formed in the early 1990s (see Figure 2-3). With the growing capacities of software, computer aided technologies such as knowledge processing and computer simulation have been used together with CAD in

different product lifecycle stages to support product design, planning, and manufacturing since the 1980s. The communication among different software used at various stages of the product lifecycle has become an important consideration for increasing the data exchange efficiency and enhancing the performance of product development and manufacture. Many approaches have been put forward to improve the data exchange, such as the interfacing approach, standardised data format or common databases (Krause *et al.*, 1993). However, these standardised formats could only represent two- and three-dimensional geometrical models at that time. Therefore, the demand to develop a unique and consistent system-independent representation of a product that also enables data management of all relevant product information led to the formation of the concept of computer-aided product modelling.

2.2.5 Product Model

In this thesis, a product model is defined as an information representation that provides data contributing to build the form (geometry and topology), function (intent) and behaviour (load resistance, etc.) of a product in a modelling process (Tolman, 1999). It is employed throughout the entire lifecycle of a product to structure product data and design information.

Therefore, a product model that can represent all detailed design information will enable designers to work on the design tasks without previous design experience. Time and cost on tutorial sessions and training for new designers could be saved. Also, a product model that captures all required design knowledge can perform as a knowledge base for different elements, varying from originality to final product. In this manner, even a complex product, which has features and structures that are special and cannot be modelled or physically produced in a straightforward way, can be represented by a comprehensive model.

Companies and industries can achieve more sales and high revenues by offering multiple product variants and options to customers. Design automation in product development is an impactful differentiator among business competitors as a reduction in time and manpower

cost of a product will allow companies to make flexible business strategies and be price competitive in the market. In other words, design automation enables companies and industries to deal with customised products more quickly and efficiently by automating repetitive design tasks in the existing product design processes. The next section discusses the Design Engineering Automation technique and product models used in Design Engineering Automation.

2.3 Design Engineering Automation

Design Engineering Automation is generally referred to as the reuse of engineering knowledge to perform a design task in an automatic way. To support design engineering by implementing and reusing knowledge in solutions, tools, or systems, Cederfeldt and Elgh (2005) pointed out that two aspects need to be considered in design automation: information handling and knowledge processing. In this research, information handling can be described as the reuse of CAD models, and knowledge processing refers to the reuse of existing knowledge, for example, rules and constraints; both aspects are incorporated to produce design variants.

2.3.1 CAD Models (Geometric Models)

As mentioned in the previous Section 2.2.1, CAD has been introduced in product design since the 1960s as a design automation technique to help reduce errors and time spent on tedious design tasks and accelerate the design process. By the year 2020, numerous CAD tools and software were developed by the CAD industry that allowed users to build 2D or 3D computer models of configured products. According to the global CAD software market research report (Research and Markets, 2020), 3D CAD software is being widely adopted because of the rising number of professionals in design engineering fields. There are mainly three types of 3D CAD models: wire-frame, surface and solid.

a) Wire-frame model

A “wireframe” is generally defined as a collection of curve segments that represent an object’s edge (Tilove, 1981). A wire-frame model represents the shape of a solid object by the network of vertices. In other words, the geometry of a product is described by its characteristic line and points. A wire-frame model is the least complex model for representing a 3D object; however, using a wire-frame model to represent a product is sometimes ambiguous (Kellie, 2010) and verbose since users need to use a lot of low-level data to describe a simple geometry such as a cube (Requicha and Voelcker, 1982). A wire-frame model cannot represent an actual solid object because it does not define the volume or surfaces of an object.

b) Surface model

A surface model is a 3D shell object with infinitely thin walls that do not have mass or volume (Butorina and Vasilieva, 2018). It is used mainly to describe complex or specialised surfaces and provide external aesthetics of a product such as turbine blades, car body panels, boat hulls and aircraft fuselages (Cam *et al.*, 1983; Yip-hoi, 2011). The surface model eliminates the ambiguity in the wire-frame model by defining adequate data on a product’s surface and by hiding lines not seen, but it provides no information about the inside of an object which is similar to the wire-frame model.

c) Solid model

A solid model is generally referred to as an unambiguous computer representation of a physical solid object created through solid modelling (Requicha and Rossignac, 1992). It uses topological information in addition to geometrical information to represent the object. Based on Shapiro’s research (Shapiro, 2002), a solid model is developed according to the following principles:

- It should be valid and correspond to some real physical objects.
- It should represent the corresponding physical object unambiguously.
- It should support any geometric queries that may be asked of the corresponding physical object.

The solid modelling technique will be further discussed in Section 2.4. A solid model contains metrics and dimensions of the solid object and invisible topological information such as the connectivity, neighbourhood, and relationship, etc. A solid model is different from a surface model, even if they may look the same on-screen. For example, a solid model can be cut and sliced open, while a surface model cannot because the surface model is hollow. Additionally, a solid model must be modelled geometrically correct. In contrast, a surface model could be geometrically and physically incorrect but still appears correct for visualisation because no properties of mass and thickness (volumes) are defined in the surface model.

2.3.2 Automation with CAD Models

In modern engineering and product development scenarios, the application of previous designs and processes for the new generation of product variants has become a common and essential factor (El Hani, Rivest and Maranzana, 2012). In the CAD domain, automation refers to the reusability of the CAD models where CAD data can be used or adapted to different designs with minimal effort (Camba, Contero and Company, 2016). The implementation of parametric modelling to develop CAD models has accelerated the product development process by allowing designers to reuse and make alterations to existing models in an efficient and easy manner.

Parametric modelling is a modelling method with the ability to change the geometry of the model when the values of dimensions are changed. In parametric modelling, the geometric model is developed and controlled by non-geometric features called parameters, which can be

further defined by dimensional, geometric, or algebraic constraints (Camba, Contero and Company, 2016). The key advantage of parametric modelling is that, for modifying the existing models or developing product variants, there is no need to redraw or recreate the model from the beginning. The previous parametric models can be reused by changing the dimension to achieve the new desired model. Moreover, parametric modelling also allows users to create the algorithm of the model, which includes rules that occur when parameters are defined and associated with each other. These rules can be further used for automation as a restrictor or limit that determines the boundaries of an event (Kalkan, Okur and Altunışık, 2018).

Parametric modelling is available by using parametric tools in most modern CAD tools. However, these parametric modelling tools are of limited use to users who do not understand the design principles or do not have a solid 3D modelling foundation (Rynne and Gaughran, 2007). In the industry, to overcome this problem, internal CAD models and guidelines are usually developed by previous/experienced designers and followed by all designers as templates to ensure the quality of the modelling and standardisation. However, this solution is restricted only to industry settings and is not applicable to general scenarios. The capability of creating robust and reusable CAD models strongly depends on the user's cognitive abilities and skills to understand and break down the design (Rynne and Gaughran, 2007).

The automation with parametric CAD models has shown significant influence (G. P. Gujarathi and Ma, 2011; Kedar *et al.*, 2018; David, 2019) in the modern CAD domain in terms of reusing parameters to generate the geometry; however, the information stored in the CAD model is still limited to geometric data and parametric values. The associated information, such as design intent and design rules, is not contained in the model and needs to be provided separately.

2.3.3 Extension of CAD Models

Although parametric CAD models do not include design intent or design rules, the parametric modelling method itself shows a potential way of enabling the addition of design semantics to a CAD model. By editing some parametric values, design semantics could be translated into the modification of existing models (Camba, Contero and Company, 2016).

Many efforts have been made by earlier researchers to extend the CAD models by integrating CAD with CAE information. Knowledge-intensive CAD (KIC) is one of those early concepts designed to address the issue of information exchange between various stages of product development by taking advantage of knowledge flow (Tomiyama, Mäntylä and Finger, 1995). It focuses on integrating design lifecycle and engineering knowledge with CAD. Shephard et al. (2004) proved that CAD models could be used for simulation-based design under a controlled interactive design and analysis environment by simplification and data management. Van der Velden (2007) pushed the integration of CAD and CAE a step further by developing a Graphical User Interface (GUI) based system that could manage the propagation of changes in CAD and change analysis along with meshing of the entire CAD model without any geometry simplification. Xu and Chen (2009) pointed out that one challenge of integrating CAD and CAE is to manage complex products that have more detailed information. Another challenge of implementing KIC is the mismatch between the availability of information and the accessibility of the appropriate information to designers (Blessing and Wallace, 1998). Blessing and Wallace (1998) published an innovative method of collecting and indexing knowledge based on context. They developed a process-based support system (PROSUS) that provides designers with accessible and relevant life cycle knowledge. All the captured knowledge in PROSUS can be further reused by other project partners.

2.3.4 Product Data Management and Product Lifecycle Management

Large amounts of product information within different formats are generated at industrial companies every day (Peltokoski, Lohtander and Varis, 2015). The increased amount of product variants also leads to the increased complexity of product information. Product Data Management (PDM) is a system that helps manage product data created during the design processes (see Figure 2-4). The PDM system was first employed as a file-based system that allows designers to save and retrieve their drawings and bills of material (BOM) (Wilson, 2006). Nowadays, the PDM system has been widely employed in industry to deal with the increased amount of product-related information such as engineering drawings, geometry data, part and assembly files, test and analysis data, BOM, etc. (Gao *et al.*, 2003; Könst, La Fontaine and Hoogetboom, 2009). Könst *et al.* (2009) pointed out that it is also important to save information like “why a decision was made”, “what solutions were used” into the PDM system; however, all these kinds of knowledge are often experienced-based and in heads of the designers and also hard to capture and store during the design process. Abramovici *et al.* (1997) assert that the aim of PDM is to serve as a product data repository which is accessible to all designers. PDM itself cannot deal with product lifecycle information, and this is where Product Lifecycle Management (PLM) comes in. PLM can be regarded as an extension of PDM that handles the information of workflows after the product development process in the product lifecycle, for example, shipment, maintenance, customer service, etc. (Peltokoski, Lohtander and Varis, 2015).

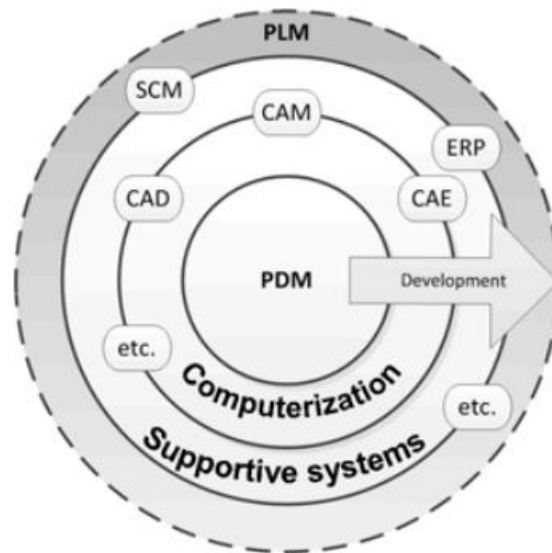


Figure 2-4: Development of PDM. Source: Moorthy and Vivekanand (2007)

While PDM only focuses on design data management relevant to the product development process, PLM focuses more on product development and manufacturing processes related to product lifecycles. PDM relies on the improvement of product data management to improve the efficiency of the existing product development process. In contrast, PLM uses PDM along with other technologies, such as Supply Chain Management (SCM) system (Bouhaddou *et al.*, 2012) and Enterprise Resource Planning (ERP) system (Avvaru *et al.*, 2020) to manage product lifecycles and boost production.

2.4 Product Modelling Methods

In recent decades, industries have recognised that traditional CAD is only able to retain the design results. Neither the design intent nor the methods will be captured or incorporated when using traditional CAD (Cooper, Fan and Li, 2001). Product modelling plays an essential role in product development activities, as it “generates an information technology reservoir of complete product data to support various activities at different product development phases” (Krause *et al.*, 1993). The method to use in the product modelling depends on design problems, the designer’s knowledge, experience and the requirement. In the literature, various product modelling methods have been developed for the realisation of

different product models. Yang et al. (2008) categorised product modelling methods into four categories: solid product modelling, feature-based product modelling, knowledge-based product modelling and integrated product modelling. Since integrated product modelling can be considered as a combination of the former three methods, three main types of product modelling methodologies are reviewed in this section.

a) Solid product modelling

Over the last ten years, the term “solid modelling” has been associated with using CAD systems to create the shape and form of product geometry and associated physical properties for the purpose of engineering design automation (Chang, 2015). Solid product modelling (Shapiro, 2002) is a technique that uses mathematical principles and computer modelling to achieve precise representation of three-dimensional objects. It is now a mature tool widely implemented in the product modelling field (Yang *et al.*, 2008). There are two standard methods of solid product modelling: boundary representation (B-rep) (Stroud, 2006) and constructive solid geometry (CSG) (Shapiro, 2002). In the B-rep method, the product is divided into a number of faces bounded by edges. In turn, the edges are bounded by two vertices at last. The B-rep method provides a fast display of a product geometry with basic information about the faces, edges and vertices (Stroud, 2006). The CSG method breaks the product into a binary tree of basic solids, for example, cylinders, spheres, cones and cubes etc. The product itself in CSG is considered as a combination of those basic solids by utilising union, difference and intersection operations (Yang *et al.*, 2008).

Therefore, it can be seen that both B-rep and CSG present a clear and simple data structure of a product. However, Chen and Wei (1997) pointed out that the weakness of solid product modelling lies in providing all necessary information for an entire product development lifecycle. Solid product modelling works in a different way from a human product designer because it can only create models with basic geometric information such as dimension,

tolerance etc. For a complete product lifecycle, more necessary information is still required, for example, “how the product will be manufactured”, “what the function of the product is”, etc.

b) Feature-based modelling

Feature-based product modelling (Chen and Wei, 1997) is seen as a well-developed extension of solid product modelling. The “Feature” here is defined by Salomons et al. (1993) as information sets that refer to aspects of form or other attributes of a part. These sets can be used to reason the design, performance or manufacture of the part or assemblies they constitute. Chen and Wei (1997) mentioned that feature-based product modelling shows great advantages over conventional solid modelling methods, such as capturing design intents, relating functionality with product geometry, working on high-level shapes instead of geometric details, etc. In the past twenty years, much research has been conducted using features to support product design (Michael J. Pratt, 1988; Wingård, 1991) and the assembling process (Van Holland and Bronsvort, 2000). However, a product from feature-based modelling is not able to transfer knowledge such as expertise and experience to other designers.

c) Knowledge-based product modelling

To automate repetitive design activities, the product needs to be modelled in a way that the model can be computed automatically and then re-generated through programmes by automating design routines. The methodology that provides a combination of object-oriented programming, Artificial Intelligence techniques and computer-aided design technologies (Chapman and Pinfold, 1999) is known as knowledge-based engineering or KBE. It aims to reduce the time and costs of product development by automating repetitive design tasks and optimising the design process in all aspects of the design process.

Knowledge-based product modelling is characterised by capturing and reusing engineering knowledge such as human expertise and product and process knowledge in the modelling process. In 1989, Lawrence et al. (1989) suggested a public recognition of the potential of knowledge-based engineering - “Although it’s not yet widely known, knowledge-based engineering is having a profound effect on how a few companies are speeding their products to market”. In the 1990s, Salustri (1996) proposed a formal theory for knowledge-based product model representation known as the axiomatic information model for design (AIM-D). It aims to provide a strictly logical framework for specifying information about a product at any point during its development. Jurit et al. (1990) integrated frame-based representation and rule-based representation into a feature-based modelling system. It can be seen as an early attempt to combine knowledge and expertise into a product modelling system. In the last 20 years, KBE has shown its advantages in product development in different engineering areas such as automation, civil engineering and aerospace engineering in terms of modelling (Rosenfeld, 1995) and cost-saving (Reddy, Sridhar and Rangadu, 2015). By investigating a range of KBE projects, Reddy et al. (2015) reported statistics of a number of KBE application results. From Reddy’s work, it can be clearly seen that time and manpower costs are significantly reduced at the design stage, resulting from knowledge reuse through KBE.

This section introduced the different product modelling methods that existed in the literature for the development of a product model. The following section explores the product model development concerning knowledge representation in recent decades.

2.5 Product Model Development

The use of KBE methods and techniques has played an important role in design engineering automation for the development of a product in the industry (Shehab and Abdalla, 2001; Sanya and Shehab, 2014). As discussed in Section 2.2.5, a product model that represents and captures all required design information could be deployed as a knowledge base that provides

guidance for design engineers to work on the design tasks without previous design experience. Cooper et al. (2001) put forward three key features of a knowledge-based product model: the What, the How, and the Why of the design. The “What” provides definitions of a product, such as its shape, components, configurations and features. The “How” refers to the sequence of steps, actions and transformations required to derive a product configuration based on input requirements. The “Why” reveals the design intent behind the single rule and the chain of reasoning that led to the final design outcome.

Many research works have been done to provide a generative representation of a product (listed in Table 2-1). Isaksson et al. (2000) provided a brief content structure for a general product model for product modelling (shown in Figure 2-5). The product data will vary depending on the availability of data, length of the product life cycle and complexity of the product, etc. It was also highlighted that the development of digital product models will reduce the need for physical prototypes and help share and reuse product data throughout the product life cycle.

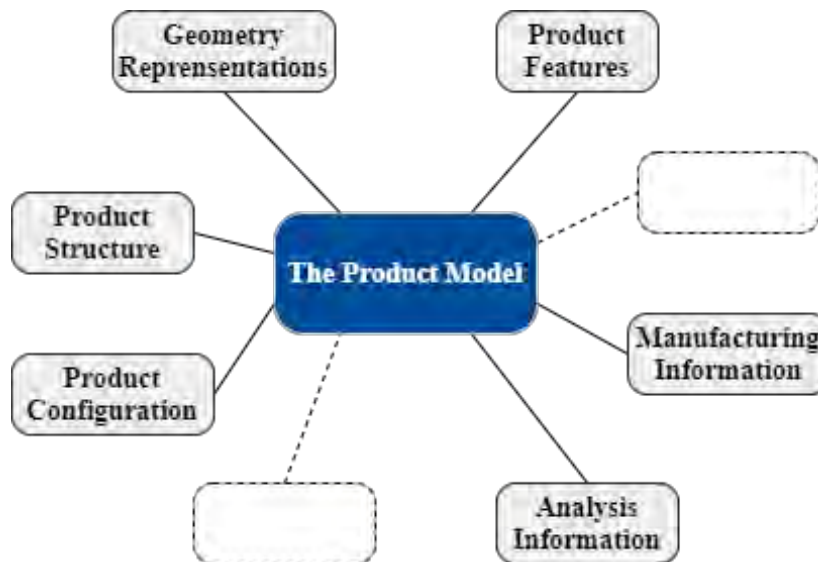


Figure 2-5: Possible content of the Product Model. Adapted from Isaksson *et al* (2000).

The Core Product Model (CPM) (Fenves, 2001) is one of the most acknowledged product models for representing design information in the literature. CPM focuses on the

representation of product model data, including function, form, behaviour and material, physical and functional decompositions, and relationships among these concepts. It has been successfully used in different Engineering projects (Roy, U., R. Sudarsan, R. D. Sriram, K. W. Lyons, 1999; Szykman, S., 1999). However, in the literature, this CPM is mainly used as a data model for representing content-level design information. There is a lack of substantiation that provides detailed implementation steps for applying this Core Product Model through use cases and tools. The interaction between the product model geometry and the design information remains unclear in the Core Product Model.

Table 2-1: Existing product model development work to support generative representation

Model or system developed in literature	Adopted methods	Reference
Featured-based and rule-based knowledge representation	Represent knowledge using feature	(Jurit H., Saia, A. and De Pennington, 1990)
Frame-rule structure for mould product design system	Represent knowledge using numbers of frames related to each other by relationship	(Lou, Jiang and Ruan, 2004)
Axiomatic information Model for Design (Aim-D)	Formal basis and logic of product structure	(Salustri, 1996)
Core Product Model (CPM)	Non-geometric information class	(Fenves, 2001)
Product Family Evolution Model (PFEM)	Extend CPM by adding rationale for design changes	(Wang <i>et al.</i> , 2003)
Open Assembly Model (OAM)	Extend CPM with assembly relations	(Mehmet <i>et al.</i> , 2005)
A unified central product model in the Unified Modelling Language (UML)	Extend CPM with assembly relations	(Gross <i>et al.</i> , 2009)
A simulating UML model of the FireSat mission satellite	UML classes with CAD model	(Gross and Rudolph, 2012)
Multi Model Generator for Aircraft Design	UML KBE Techniques	(Rocca, 2011)

A knowledge reasoning engine can be used to realise the inference mechanism to provide the “How” and the “Why” for solving a posed problem in industry (Chiang, Trappey and Ku, 2004). As explained by Rocca (2011), the knowledge reasoning approach in the inference mechanism can be performed by selecting, using and matching various rules. Isaksson (2000) pointed out that it is important to define rules within the model to provide an interpretation for both humans and computers to avoid misunderstandings and misuse of product models. The rules are usually defined through the use of object-oriented language to handle semantics and internal relations within the model. With the development of object-oriented techniques, many languages have been adopted to handle rules in product model development. EXPRESS (ISO, 1994) has been designed to describe product information under the Standard for the Exchange of Product model data (STEP) (ISO 10303-1:1994, 1994). The Unified Modelling Language (UML)/System Modelling Language (SysML) was a visual language generally used in system development to display and describe the structure of systems. C# is an object-oriented and component-oriented programming language designed by Microsoft to support the development of robust applications where rules are programmed and executed. All these languages are successful in modelling rules in their domains, and the choice of an appropriate object-oriented language depends on the user’s preference based on the consideration of the development environment and the availability of tools.

Engineering rules are often formed from product design and process knowledge (Melody Stokes, 2001). Rocca (2011) mentioned that knowledge could be stored in the form of rules to provide the inference mechanism for solving problems. In Rocca’s work, the design rules for product modelling are defined into the following five different types:

- Logic rules – IF-THEN-ELSE rule and complex conditional expressions
- Math rules - mathematical rules including trigonometric functions and operators for matrices and vectors algebra, etc.

- Geometry handling rules - the generation and manipulation of geometric entities and parametric rules
- Configuration selection rules – a combination of mathematical and logic rules to change and control the topology of the product model.
- Communication rules – specific rules that allow data communication and interaction with other applications.

After reviewing state of the art in the product model development, it can be seen that the development of a knowledge-based product model requires the fulfilment of the following aspects:

- Providing a generative product representation that can provide all associated design information for product modelling
- Providing the capability of knowledge reasoning with the captured design rules

Current research works show that design rules from the existing product design knowledge can be used to build the interaction between geometry and design information and thus provides the knowledge reasoning. The next chapter will explore the knowledge-based engineering techniques, product modelling standards and tools to provide a deeper understanding of how to capture and reuse product design knowledge for developing a knowledge-based product modelling framework.

2.6 Chapter Summary

The chapter provides an overview of product design and modelling with an understanding of the product design process for product development. This chapter further discussed the advancement of product design with computer-aided technologies and presented a clear understanding of DEA. It also highlighted the existing product models and techniques presently available for product modelling for DEA. Also, different existing product

modelling methods were discussed. Based on the findings from the literature review, it was identified that KBE methods were needed to support the design engineering automation for the development of a product model. This leads to the next chapter, which further discusses the state of KBE methods and techniques for capturing and reusing knowledge and introduces product modelling standards and tools.

3 Knowledge-Based Engineering Techniques, Product Modelling Tools and Standards

3.1 Introduction

This chapter discusses the different KBE techniques to provide a clear understanding of KBE methods used to capture and reuse knowledge in the product modelling process. Knowledge Capture Methodology (KCM) is identified as the best applicable KBE technique. In addition, this chapter also discusses model-based engineering and model-based system engineering concepts to explain how a product can be decomposed into modular components and how the product data can be exchanged between different platforms. Next, product modelling standards and tools are discussed to identify the most suitable interoperable standard and development tools for this research. Further, research gaps are identified based on the findings from the literature review in Chapter 2 and this chapter.

3.2 Knowledge-Based Engineering (KBE)

KBE is a relatively new engineering method (since 1992) that provides a combination of object-oriented programming, Artificial Intelligence techniques and computer-aided design technologies (Chapman and Pinfold, 1999). KBE systems were developed to capture the product and process information to support the modelling of engineering or business processes. The resulted model from KBE systems could be used to automate all or part of the process, which will shorten the development of the product and help deliver the design faster (Chapman *et al.*, 2007). By explaining KBE from the perspectives of KBE stakeholders (a company manager, a KBE developer, Engineers, Users, etc.), Rocca *et al.* (2012) presented that KBE is a high potential technology to support product design through knowledge reuse and design automation.

3.2.1 KBE and CAD

A misunderstanding that often arises with KBE is that it is an alternative to CAD. Cooper et al. (Cooper, Fan and Li, 2001) presented in their work that KBE does not replace the need for CAD, but it will help reduce the CAD activities that are needed for a particular task and free up design engineers for other programmes. CAD will still be required to provide geometry files to relate KBE application. Since KBE techniques are identified as enabling methods that could be utilised to achieve knowledge capturing and reusing in the product modelling process, a range of KBE methodologies is reviewed in the following Section 3.2.2 to understand how they deal with knowledge capture and reuse and to identify the best applicable KBE method.

3.2.2 KBE Methods

MOKA (Methodology and tools Oriented to Knowledge-based Applications) (Melody Stokes, 2001) is one of the most successful KBE methodologies for capturing, structuring and formalising knowledge in recent years. It splits the knowledge up and associates different knowledge with predefined problem domains, allowing users from diverse technology backgrounds to select and use it (Reddy, Sridhar and Rangadu, 2015). MOKA divides the KBE application lifecycle into six phases (as shown in Figure 3-1).

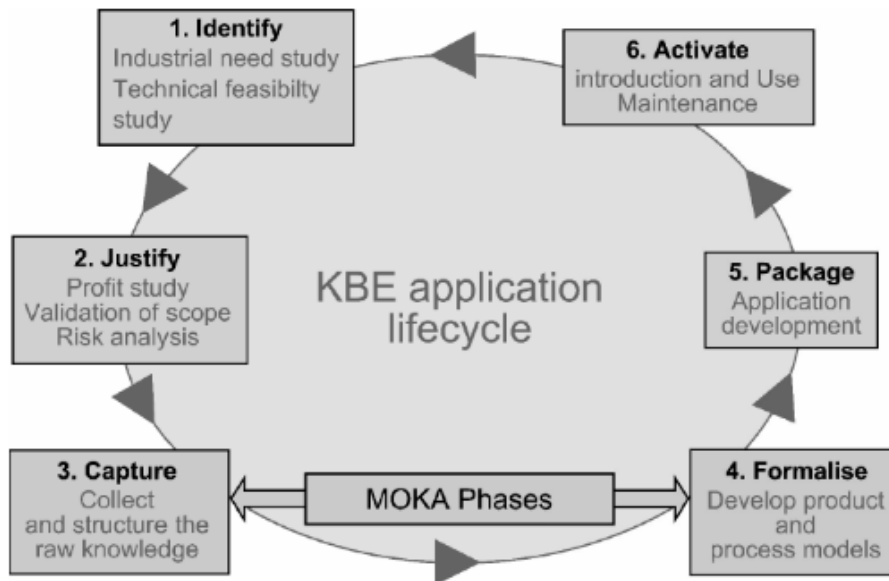


Figure 3-1: KBE application lifecycle from MOKA. Source: Melody Stokes (2001).

MOKA methodology emphasises the capture (phase 3) and the formalising of the structured knowledge (phase 4) through the use of informal and formal models. As explained by Oldham, the “Capture” phase uses ICARE forms (Illustration, Constraint, Activity, Rules and Entity forms) as an “Informal Model” to collect and structure the existing knowledge (Oldham *et al.*, 1998). In this way, the knowledge is represented so that users can understand it without being experts in formalisation languages. The “Formalise” phase converts the captured knowledge from the “Informal Model” into a “Formalised Model” using an interoperable format that can be used by digital tools and software. MOKA has been regarded as a successful methodology for the development of a KBE system (Gómez de Silva Garza and Maher, 2000). It addresses two main focuses on the developing KBE system for design engineering automation: “a common and unifying framework” and “the reuse of knowledge” (Klein, 2009).

While MOKA was aimed at larger industrial KBE applications and relied on heavily industrial involvement, Lovertt *et al.* (2000) proposed a methodology to support KBE

implementation in Small to Medium Enterprises (SMEs), namely KOMPRESSA (Knowledge Oriented Methodology for the Planning and Rapid Engineering of Small-Scale Application). Unlike MOKA, KOMPRESSA can satisfy end-user requirements through the entire development process of KBE with the involvement of existing techniques and experience from the enterprise to a great extent (Chapman *et al.*, 2007). In KOMPRESSA, five key requirements are listed for developing a KBE application: functionality, user interface, information, knowledge elicitation and performance (Lovett, Ingram and Bancroft, 2000). The KOMPRESSA modelling set is shown in Figure 3-2. KOMPRESSA shows the capability of using graphical diagrams and supplementary text to capture knowledge from end-users, although limited tools and techniques are proposed.

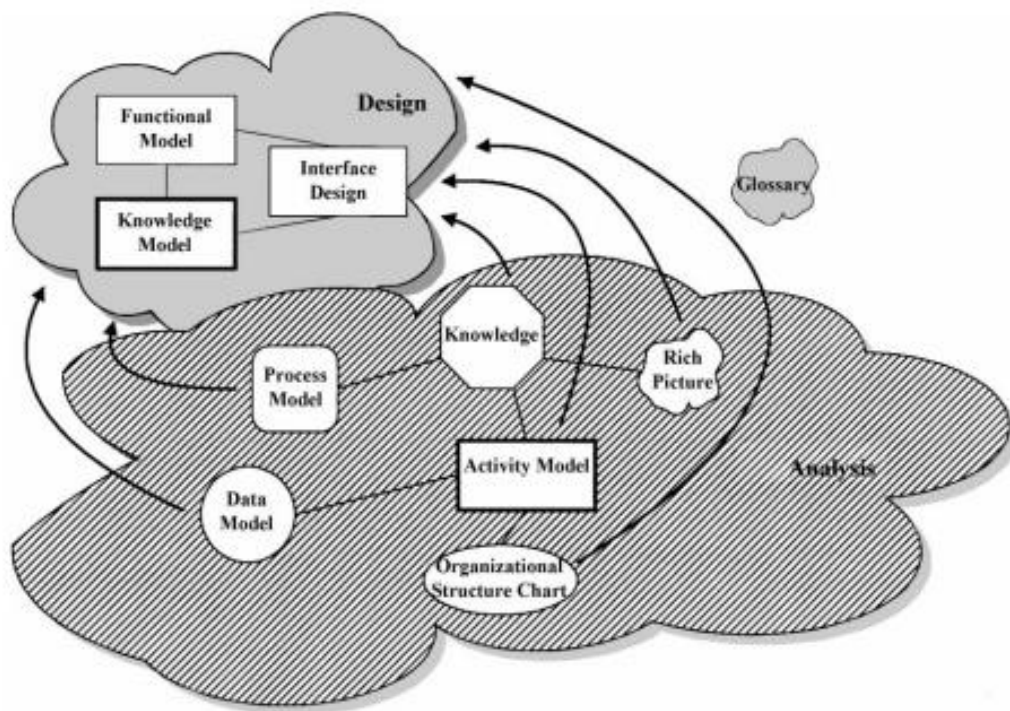


Figure 3-2: The KOMPRESSA modelling set. Source: Lovett, Ingram and Bancroft (2000).

DEE (Design and Engineering Engine) (Rocca and Tooren, 2007) is an overall multidisciplinary design optimisation approach that is more powerful than MOKA due to its advanced analysis mechanism and optimisation methods (Reddy, Sridhar and Rangadu, 2015). It extends MOKA by providing detailed multidisciplinary design analysis and

optimisation in the KBE application. The DEE methodology process flow is shown in Figure 3-3. However, Curran et al. (2010) indicated that the limitation of DEE exists in the lack of methods for knowledge acquisition and transition.

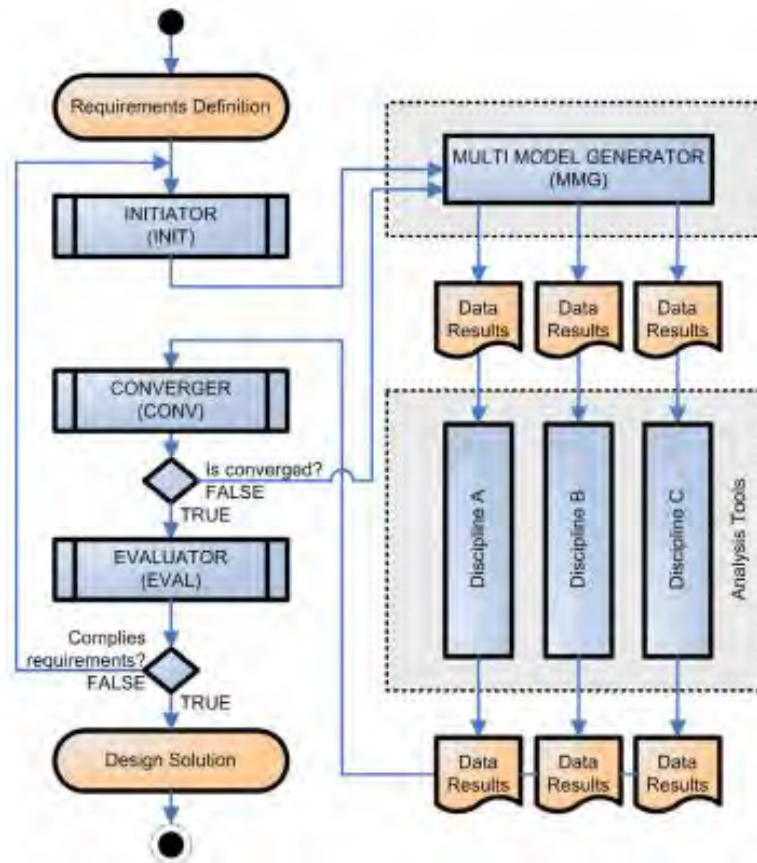


Figure 3-3: DEE process flow to support multidisciplinary design optimisation. Source: Berends, Van Tooren and Schut (2008)

KNOMAD (Knowledge Nurture for Optimal Multidisciplinary Analysis and Design) (Curran et al., 2010) is a KBE methodology developed for multidisciplinary analysis and design by utilising integrative knowledge in product design and manufacture. It was developed to address the shortcoming in DEE and to support the requirements of knowledge maintenance that are not available in MOKA. Figure 3-4 shows the KNOMAD Methodology framework.

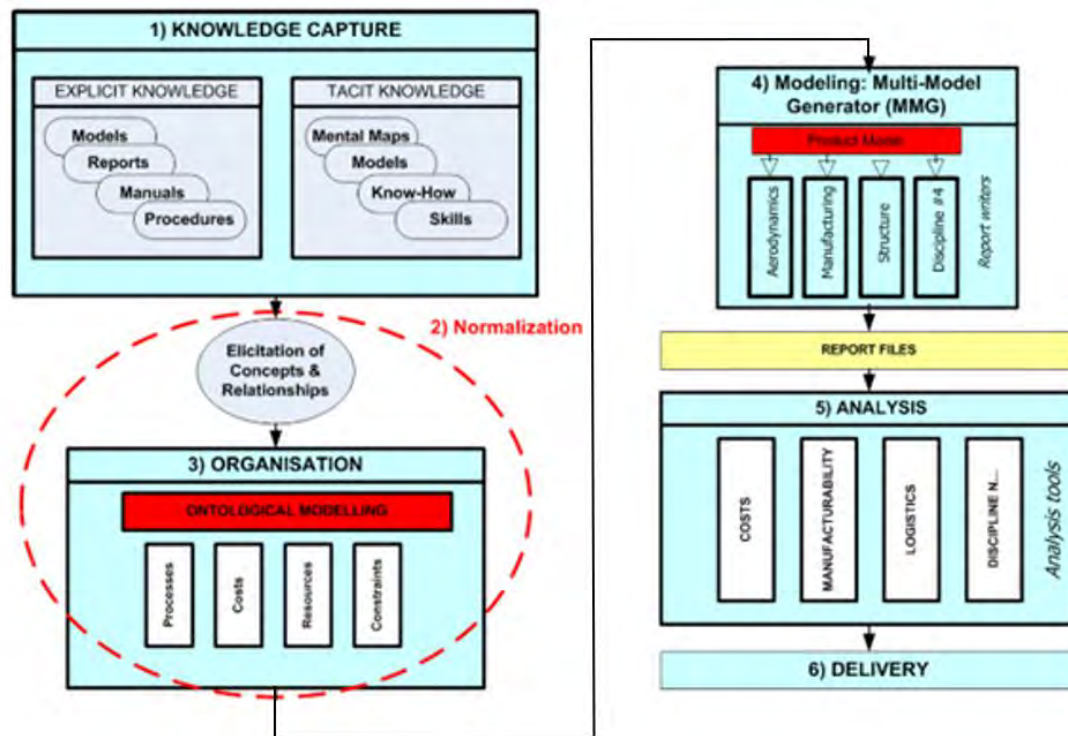


Figure 3-4: The KNOMAD Methodology. Source: Curran, Verhagen and Van Tooren (2010).

Both DEE and KNOMAD are targeted for multidisciplinary knowledge representation to improve the current KBE process. However, there is still a lack of substantiation for the individual methodology steps with respect to implementation for product modelling.

KCM (Knowledge Capture Methodology) (Chapman *et al.*, 2007) focuses on capturing design knowledge and decomposing a product into parts with attributes and reorganised knowledge. After classifying subcomponents and building relationships between them, parameterisation is performed based on the application environment and constraints. As a result, subcomponents can be reused in new product development (Terpenney, Strong and Wang, 2000).

In KCM, there are eleven steps that guide the users from understanding the product model to the decomposition of the product model and identifying the part, attribute and knowledge that meet the requirements within a visualisation environment. Figure 3-5 shows a diagram of the eleven steps provided by Terpenney, Strong and Wang (2000)

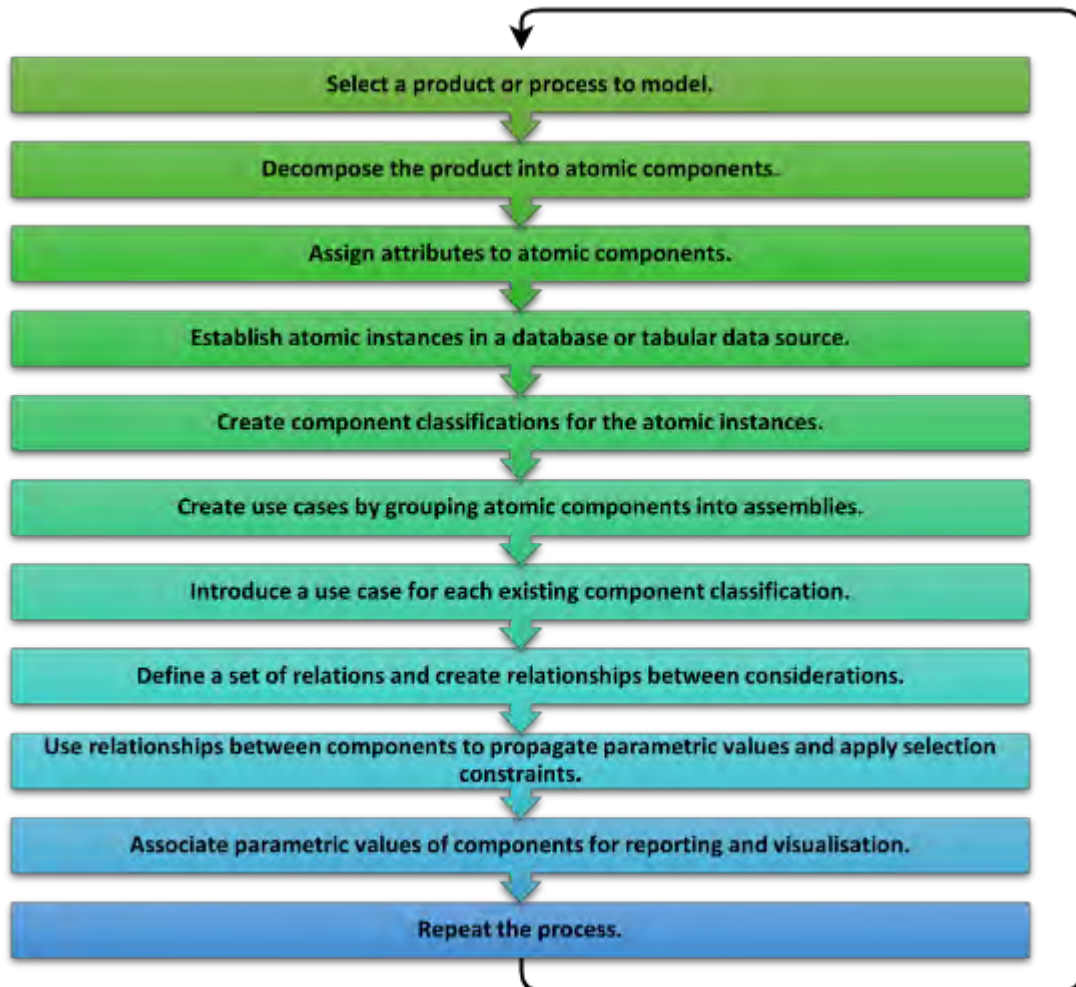


Figure 3-5: The eleven KCM steps. Adapted from Terpenney, Strong and Wang (2000).

According to Chapman et al. (2007), KCM is an appropriate method for product modelling as it has evolved from a natural understanding of the way design engineers work and also presents clearly how parts and their relationships to a product are generated from a designer's perspective. KCM is suitable for parametric geometry representation as components are associated with parametric values in the KCM process (Terpenney, Strong and Wang, 2000).

All these reviewed KBE methodologies shared many main principles with MOKA and have been created to serve different areas. Table 3-1 shows the focus areas of different KBE methodologies.

Table 3-1: Focus areas of reviewed KBE methodologies

KBE methodologies	Focus area
MOKA	Capture and formalise knowledge for larger KBE application
KOMPRESSA	KBE application development for SMEs
DEE	Multidisciplinary design optimisation
KNOMAD	Multidisciplinary analysis and design
KCM	Design knowledge capture and decomposition

By comparing the methodologies introduced above, it can be seen that KCM has greater compatibility with product modelling, especially in capturing, structuring and decomposing design knowledge. Because in KCM, a product and its design knowledge are decomposed into components. Each component is linked with an instance in the database. By defining the relationship based on design rules and customer specifications, a component can be reused and associated with new parameters in a new model. The knowledge capture and product decomposition processes can be repeated to provide a comprehensive understanding of necessary knowledge from each component of a product for reuse (Chapman *et al.*, 2007). The implementation of the KCM methodology in this research is further explained in Chapter 5.

Although these reviewed KBE methodologies have shown advantages in terms of knowledge capture, structuring, and reusing, they still have some limitations. Studies from Fan and Bermell-Garcia (2008) and Cederfeldt et al. (2006) show that there are “black-box” problems in the communication between different KBE systems. Performed tasks and processes by the KBE systems are implemented in a way that is not readable and understandable to the end-users. The transparency of KBE systems is necessary to provide adaptable, structured and reusable knowledge bases. Furthermore, Curran et al. (2010) pointed out that one major shortcoming of existing KBE methodologies is the lack of substantiation steps. Limited

implementation advice and use cases with tools and techniques examples are provided and discussed in the existing KBE methodologies.

All these limitations need to be addressed in this research for the development of a product model and for the development and implementation of a KBE framework for the purpose of capturing and reusing design knowledge in product modelling to support design engineering automation. In agreement with Silva (2015), a model itself can be regarded as a system, with its own identity, complexity, elements, relations, etc. Meanwhile, the implementation of the KBE framework requires the development of a prototype KBE system. Therefore, in this research, the development of a product model and the development and implementation of a KBE framework can be considered from a Model-based Engineering view. The following section explores KBE with Model-based Engineering to provide an understanding of the development of a product model and to explore the methods of avoiding “black-box” problems for the development and implementation of a KBE framework.

3.3 KBE with Model-Based Engineering

3.3.1 Model-Based Engineering Overview

In System Engineering, Model-based Engineering (MBE) is specialised in aspects including system architecture, requirement traceability, performance analysis, simulation, etc. MBE is “an approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and product throughout the acquisition life cycle”(NDIA Systems Engineering Division M&S Committee, 2011). It is an emerging approach used to deal with the increasing complexity of systems.

MBE is commonly classified as the model-based system engineering (MBSE) in the System Engineering field. The International Council on Systems Engineering (INCOSE) defines

MBSE as the “formalised application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” (International Council on Systems Engineering, 2007). It helps to improve interactions of stakeholders and developers by providing better knowledge capture, domain information sharing and enhanced reuse of artefacts (Rodrigues Da Silva, 2015).

3.3.2 Applying Key Concepts from MBE and MBSE into KBE

A product model itself can be regarded as a system. In KCM, a product model is decomposed into atomic models with attributes and reorganised knowledge. This can be regarded as a system with sub-systems under it in MBE. To deal with models and subdivided models in KCM, some key concepts in MBE and MBSE are adapted in this research.

a) UML/SysML structure

A standardised and robust modelling language is the key enabler of MBSE (Soyler and Sala-Diakanda, 2010). In order to make models more visual and intuitive, software and systems engineers have developed various graphical modelling languages.

The Unified Modelling Language (UML) is a general-purpose graphical modelling language that can be used to specify, visualise, construct, and document the artefacts of software systems (Jon Holt, 2013). It was adopted by the Object Management Group (OMG) as a standard in November 1997 (Rumbaugh, Jacobson and Booch, 1999). The UML defines thirteen types of diagrams that capture decisions and understanding about a system and allow the requirements, behaviour and structure of a system to be defined. UML has been widely used among multiple application domains (Object Management Group, 2015).

The System Modelling Language (SysML) is an extension of UML that originated by adapting the UML for System Engineering applications (Jon Holt, 2013). It provides an additional set of modelling diagrams and constructs to model complex systems, including

hardware, software, data, procedures and other system components (Estefan, 2008). In particular, as described by OMG, SysML provides graphical representations with “a semantic foundation for modelling system requirements, behaviour, structure, and parametric, which is used to integrate with other engineering analysis models” (Object Management Group, 2008). Together, UML and SysML provide designers and users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models. Although UML and SysML are initially developed for System Engineering use, they are not attached to any methodology or domains (Estefan, 2008). Many software engineering methodologies and commercially-offered model-based systems incorporate the UML and SysML into specific methods and artefacts produced as part of the methodology (Bajaj, 2008; Bajan *et al.*, 2011). Thus, the UML and SysML can be regarded as visual modelling languages applicable in this research to represent product model structure.

b) Neutral standard

In MBE and MBSE, standards or Application Protocols are always required to enable data exchange between different systems (Friedenthal, Griego and Sampson, 2007). Using neutral standards in the development of systems provides unambiguous concepts and practical support for the interoperability of tools. Thus, in this research, the product data and knowledge from KCM need to follow an interoperable standard and format to enable steady data exchange between different product modelling systems. The product model also needs to be developed in a neutral format for the exchange of data between different product modelling systems and related tools.

Therefore, two key concepts from MBE and MBSE are adopted in this research. UML/SysML is employed as a standardised modelling language to represent the product model structure. Meanwhile, in order to develop a product model that is interoperable between different product modelling systems and tools, a neutral standard and format will be

adapted to support the exchange of data during the whole product development lifecycle and to keep the product model in a consistent format for knowledge reuse.

The following sections explore the existing standards (Section 3.4) and tools (Section 3.5) in product modelling with KBE to understand and also identify the most appropriate standard and the available development and implementation tools.

3.4 Standards and Formats in Product Modelling

The following Section 3.4.1 reviews the existing formats and standards used for product modelling with CAD software.

3.4.1 Product Modelling Formats and Standards

The formats of the product model have co-evolved along with the development of CAD software. Generally, the formats of CAD files can be divided into two types which are native and neutral. The native formats are based on and created by specific CAD software. These native formats are regarded as copyrighted intellectual property of the CAD software for which they are used. Therefore, the product data stored in native formats are only readable by the respective programs. The exchange of these product data between different CAD software within native formats is performed through direct data translation. However, direct data translation is typically unidirectional, partially functional and not standardised because most native formats are proprietary (Bondar *et al.*, 2015). Examples of native formats from literature include AutoCAD (.dwg), Blender (.blend), SolidWorks (.sldprt and .sldasm), SketchUp (.skp), etc. However, since this literature review section focuses on identifying an interoperable standard/format between different product modelling systems, the review of native formats is beyond the scope of the current research.

Neutral formats are open-format and not proprietary. Neutral product modelling standards are typically used as neutral 3D formats for sharing product data between different CAD software. In this research, three neutral product modelling standards are reviewed.

a) IGES

The Initial Graphics Interchange Specification (IGES) is a neutral file format that enables digital data exchange in CAD systems. IGES is the first standardised data exchange format designed to meet the requirements of product definition data communication between different CAD/CAM/CAE systems (Zha and Du, 2002). The IGES file has the format of .igs and is based on ASCII standard code. The IGES file is written by sequence and is readable by all text editors. However, after the initial release of STEP, the development of IGES started to decline, and the last update of IGES was in 1996.

b) STEP

The word “STEP” is the acronym of the Standard for the Exchange of Product model data. It is the general name of the international standard ISO-10303, which was approved in March 1994 (Tang *et al.*, 2001). Gu and Chan (1995) described STEP as an internationally recognised standard that provides a uniform data representation and information exchange mechanism used in the product life-cycle. It also ensures a steady exchange format and dependable application interfaces between different computer systems and CAD software (Owen, 1997). The final aim of the STEP is to create a complete product representation in a general and consistent format (ISO 10303-1:1994, 1994). STEP provides a neutral way of describing information relevant to a product, all the steps of its life cycle, relying on a set of general formalisms.

c) JT

JT (Jupiter Tessellation) is the common interoperable format used across all Siemens PLM software. It became an ISO-standardized 3D data format (ISO 14306) in 2012. JT is used in

the industry for product visualisation, collaboration, CAD data exchange, and in some also for long-term data retention. It stores faceted information, boundary representation surfaces (B-Rep), Product and Manufacturing Information (PMI), and Metadata (textual attributes) either exported from the native CAD system or inserted by a product data management (PDM) system (Siemens PLM, 2019a).

Table 3-2: Comparison between IGES, STEP and JT

Standards	Advantage	Disadvantage	Reference
IGES	<ul style="list-style-type: none"> • Store drawing information in an ASCII or binary neutral format, which can be exchanged between various users easily. 	<ul style="list-style-type: none"> • Decreasing usage • Lack of formal information model • Problems during file exchanges and manipulation • Hard to understand 	(Bhandarkar <i>et al.</i> , 2000) (Marjudi <i>et al.</i> , 2010) (Zha and Du, 2002)
STEP	<ul style="list-style-type: none"> • Steady data exchange • High usage and wide dissemination • widely supported by common CAD software 	<ul style="list-style-type: none"> • Tedious classes • High costing and skills are needed for standard exchange 	(Gu and Chan, 1995) (Marjudi <i>et al.</i> , 2010) (Owen, 1997)
JT	<ul style="list-style-type: none"> • Increasing usage in Automotive • Digital mock-up (DMU) analysis 	<ul style="list-style-type: none"> • Lack of maturity in existing converters and software for data exchange 	(Fröhlich, 2011) (Siemens PLM, 2019a)

Table 3-2 shows a comparison between IGES, STEP and JT standards. By comparing these three standards in product modelling, it can be seen that STEP is the appropriate neutral standard to be used for developing an interoperable product model as it provides steady data exchange and is also widely used in industry and supported by common CAD software. The next sections 3.4.1 to 3.4.3 explore a deeper understanding of “how to model with STEP”,

and also provide an understanding of STEP Application Protocols and also review research works that have been done in product modelling by the use of STEP in recent years.

3.4.2 Modelling with STEP - EXPRESS and EXPRESS-G

EXPRESS (ISO, 1994) language is used in STEP as a tool to define the product data in an object-oriented and integrated environment (Peng and Trappey, 1998). It is an object-oriented data descriptive language by which targets are classified and built based on their data entities, attributes, relationships and constraints, etc. Moreover, it provides a mechanism to define application protocols by adding specific subclass inheriting required information from their superclass (S. Rahimifard, 1996; Peng and Trappey, 1998).

Usually, the EXPRESS information models within STEP application protocols can become quite long and complex (Kahn *et al.*, 2001). Hence, EXPRESS-G (STEP Tools Incorporated, 2008) is introduced as a formal graphical notation of EXPRESS that can assist users in understanding and managing the complexity of large information models. EXPRESS-G diagrams show relationships and structure of entities, attributes, type declarations and hierarchies of inheritance more clearly than the plain EXPRESS text. Despite this, Arnold and Podehl (1999) stated that EXPRESS-G could not reach the full expressiveness of EXPRESS. There is still a lack of possibilities to visualise functional components, local or global rules, and algorithms.

3.4.3 STEP Application Protocols - AP203, AP214 and AP242

As mentioned before in Section 3.4.1, STEP is the international standard for the exchange of product model data. It addresses product model data from mechanical and electrical design, geometric dimensions and tolerances, analysis and manufacturing, as well as additional information specific to various industries such as automotive, aerospace, building construction, ship, oil and gas, process plants and others (Curran *et al.*, 2015). Due to the complexity, STEP is divided into smaller component parts, including a series of ‘Application

Protocols' or APs, each covering a particular industrial domain. And each AP is titled by the domain that it applies to (Kc Morris, 1999).

Three Application Protocols are reviewed in terms of mechanical product design and automation: AP203, AP214 and AP242. AP203 (Configuration Control Design) (Kc Morris, 1999) applies the domain of general Mechanical CAD. It is primarily supported by the aerospace and defence industry (Ap242.org, 2016). This protocol is used to exchange data describing designs represented as solid models and assemblies of solid models.

However, the limitation of AP203 lies in the fact that it still does not cover the data which is not applied to the design phase, such as manufacturing data. In comparison to AP203, AP214 (*STEP AP203 and AP214 Protocols*, 2016) focuses more on automotive design. It contains products such as mechanical parts, assemblies, and tools used by manufacturing (in principle applied to the description of cars). The scope of AP214 (Core Data for Automotive Mechanical Design Process) in Mechanical CAD is roughly equivalent to AP203. Nevertheless, AP214 does not cover the parametric representation of shapes which is required in this research.

AP242 (Managed model based 3D engineering) (*STEP AP242 Project*, 2014) is a newly released standard in 2014 that combined the scope of AP203 and AP214 with other widely used STEP protocols. It is designed by using a modular architecture that enables further evolution and enhancements of the standard in a more flexible way. AP242 introduces a new mechanism that allows describing references between elements of several models written on separate files. This capability eases the CAD data exchange and archiving between some different protocols and improves the efficiency of processes by integrating the various enterprise functions (*STEP AP242 protocol*, 2014).

However, the implementation of AP242 in commercial CAD software is still very limited and unspecific in the literature. Some CAD vendors claim to support AP242 functionalities;

however, those are mainly the ones that have been defined in AP203 and AP214 (Coronado, 2014). Schätzle (2016) pointed out that the CAD vendors provided broad descriptions about the implementation level of AP242 only for advertising. There is a lack of detailed implementation of AP242 within CAD systems and KBE applications. The following section reviewed relevant research work in product modelling with STEP in recent years.

3.4.4 Relevant Research Work in Product Modelling with STEP

Many efforts have been made to utilise STEP to support product modelling in recent years. Gu and Chan (1995) developed a STEP-based generic product modelling (GDM) system in their research. In their work, five libraries (product and version, relationship, geometric item, material, and tolerance) are used as integrated resources to model a product. It firstly shows the capability of adapting integrated data sources to construct a product. Peng and Trappey (1998) recognised that the STEP information model has a three-layer architecture: the application, logical, and physical. The logical layer defines the product representation of entities based on the entire product life-cycle. The application layer contains the necessary models for specific applications. The physical layer provides information such as STEP file format, data exchange, etc. The three-layer architecture is regarded as guidance for developing STEP-based product modelling system architecture by other researchers (Tang *et al.*, 2001). Wu *et al.* (1992) successfully developed an information model based on STEP entities to integrate CAD and CAE applications for mechanical systems. Tang *et al.* (2001) presented a product modelling system for the stamped part and die development. EXPRESS defined schemas are used to construct the logical layer, which includes all information related to the stamped part and die, such as shape, manufacturing resource, tolerance and material. Zha and Du (2002) developed a STEP-compatible model for a mechanical product assembly planning system. They presented five fundamental product data categories for supporting a particular knowledge-based application for assembling, which are the shape information, the

form feature information, the tolerance information, the mechanical part information and the assembly information.

While the usage of STEP for product modelling was increasing, Männistö et al. (1998) stressed that the potential of STEP is restricted when companies put forward new product modelling concepts and specify more definitions. In STEP, a fixed standard data schema is predefined for modelling single products. Products of a company are also represented as instances of that schema. To be compatible with new product classes of a company, redefinition of EXPRESS instances is required to describe new company-specified concepts in terms of multiple variants of products. To overcome this problem, the company-specific specialisation of the concepts can be treated as data exchanged between different systems (Männistö et al., 1998). For example, a company can define an extension schema by using subtypes of the existing concepts in the STEP application protocol. Nevertheless, this approach may give rise to changes in the basic rules of STEP.

In the primary design stages, the descriptive information of a product is discrete and unorganised, while knowledge is in various forms instead of pure data. As pointed out by Fenves (2001), STEP is used to exchange information, which is the outcome of design activities, rather than the information generated and used through the development of a design. Therefore, STEP tends to be invoked when all design of a product is fixed and the product is ready for manufacturing.

Kim et al. (2008) mentioned that one obvious drawback of the STEP format is that it does not allow for the exchange of parameters, design intent and other data that may be associated with the CAD models. To overcome this problem, some research work has been done to enrich the product model data in STEP by mapping external data with STEP entities and classes (Arnold and Podehl, 1999; Kahn *et al.*, 2001; Barbau *et al.*, 2012). However, Barbau et al. (2012) stated that defective issues still exist in the integration of STEP data when

combined with other STEP or non-STEP product information. They presented an approach to enable the translation of STEP and its instances to Ontology Web Language (OWL) (*OWL Web Ontology Language Overview*, 2016). However, information structure translation between different languages will always generate new problems such as data missing and mismatching, and the mapping with the entire STEP standard is complicated and time-consuming.

Although many research has been done on utilising STEP for product modelling in the last two decades, Yang et al. (2008) pointed out that most of the existing research work are still very limited in terms of how to perform the integration of STEP and product modelling methodologies along with existing data resources to generate a completed product modelling framework. Therefore, to address these limitations of STEP and avoid the complex and tedious mapping process and avoid data missing and conversion errors, a new product modelling and implementation method for integrating STEP and various product data is required.

3.5 Tools in Product Modelling

During the past 50 years, various CAD software and tools have been developed to aid product design and automate product modelling tasks. As discussed in Section 2.3 of Chapter 2, this research aims to develop a method that supports DEA by reusing CAD models and capturing and reusing the existing knowledge. Thus, a range of product modelling tools has been reviewed to identify the best implementation tools which are not only capable of displaying the CAD models but also capable of capturing and reusing the existing knowledge during the product modelling process.

3.5.1 Traditional CAD Software

In this research, traditional CAD software is defined as commonly used CAD software and tools that enable users to design or visualise engineering products in 3D within an integrated graphical user interface on a computer system. Several traditional CAD software and tools are reviewed in this research (see Table 3-3) in terms of the following aspects:

- Open Source – if the software is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose.
- Geometry View – if the software can display the 3D geometry of imported CAD models in the interface
- Structure View – if the software can display the product structure (hierarchy of assembly) of the imported CAD model in the interface
- Application Development - if the software allows the users to develop applications by coding in the software
- Interface Design - if the software allows the users to develop a user interface in the software
- Data capture and reuse – if the software can capture the user’s input and store the data and reuse the data.
- Availability - if the software is accessible to users

Table 3-3: Reviewed traditional CAD software (version up to 2021) in this research

Tools	Open Source	Geometry View	Structure View	Application Development	Interface Design	Data Capture and Reuse	Availability
AML	x	✓	✓	x	✓	Only in proprietary format	License required
SolidWorks	x	✓	✓	x	x	Only in proprietary format	License required
CATIA	x	✓	✓	x	x	Only in proprietary format	License required
Siemens NX	x	✓	✓	x	x	Only in proprietary format	License required
Autodesk Inventor	x	✓	✓	x	x	Only in proprietary format	License required
Creo	x	✓	✓	x	x	x	Free
CAD Exchanger	x	✓	✓	x	x	x	Free
STP viewer	x	✓	✓	x	x	x	Free
IDA-STEP	x	✓	✓	x	x	x	Free
CAD Assistant	x	✓	✓	x	x	x	Free
Free CAD	✓	✓	x	x	x	Only in proprietary format	Free

As can be seen from Table 3-3, the reviewed traditional CAD software are capable of displaying CAD models in the user interface. However, the capability of capturing and reusing user input data is limited to proprietary format only. None of these reviewed traditional CAD software support application development or user interface design for development purposes by end-users. Most industrial and commercial CAD software, such as SolidWorks, CATIA, Siemens NX and Autodesk Inventor, are license-required.

Based on the above review, it can be seen that there are very limited tools that support the interaction between end-users and the product modelling process through the development of a user interface while using a generic format within an interoperable standard. In this research, the interaction between end-users and the product modelling process is referred to as knowledge capture and reuse of the existing design knowledge. The following Section 3.5.2 introduces Adaptive Modelling Language (AML) and discusses the availability of this knowledge-based engineering modelling tool.

3.5.2 Adaptive Modelling Language (AML)

Adaptive Modelling Language (AML) is an object-oriented, knowledge-based engineering modelling framework developed by Technosoft (Technosoft, 2013). AML application enables multidisciplinary modelling and integration of the entire product and process development cycle. AML provides the capability of user application developments which enables the users to programme in the current modelling environment. In the AML environment, a system can be integrated with multiple functions such as finite element analysis, 2D/3D sketch, etc. For these reasons, AML can be considered as a computing language to define product features, capture knowledge from the modelled domain and create parametric models with that knowledge. In addition, by implementing a user case, a product can be visualised through the integrated 3D graphical module in AML. However, due to its limited accessibility, AML is not available to general users. Thus, in the following section, alternative product modelling tools are reviewed in order to identify the most appropriate and available tool that can provide product modelling interaction, the capability of visualising CAD models and as well as capturing and reusing knowledge.

3.5.3 Gaming Engines in Product Modelling

The trend of product modelling for design engineering automation has evolved from manual drafting to CAD, from CAD to Computer Aided Product Modelling, and then to Knowledge-

Based Product Modelling. This has required the product modelling software and environment to provide more interaction between end-users and the product modelling process through the reuse of existing knowledge to support the product modelling.

Barnes (2007) stated that the development of an interactive application consists of two main components: the application and the content. The application aims to provide information in real-time to end-users and the ways to interact with it. The content contains the information through which the application navigates and provides a view to the users.

COLLADA technology (Khronos Group, 2004) has been developed and widely adopted in the domain of interactive applications in the industry. The file extension for the COLLADA format is “.dae”. Although being popular in the gaming industry, the original intention behind the COLLADA format was to serve as an international standard for the 3D visualisation of industrial data and digital assets exchange (ISO/PAS 17506, 2012). The announcement of the COLLADA format becoming an ISO standard in industrial automation systems and integration shows the potential of adopting elements from the gaming industry in Engineering. According to the interactive qualifying project report from Haas (2014), developers have recently realised that game engines can be successfully used for non-game applications development such as architecture prototyping, interactive applications and research data visualisation.

Gaming engines are generally used as an integrated development environment to enable the rapid development of game applications and build interactive applications. Although several gaming engines are in the market, Unreal and Unity are the two main gaming engines widely used in the current gaming industry.

Unreal was first released in 1995 and has become one of the major game engines being employed in the gaming industry. The Unreal gaming engine features fast rendering and high-quality graphics; hence it is preferred for building large games.

Unity is a cross-platform game engine that was announced in 2005. It has become more popular and adopted by a growing number of users in recent decades due to its easy accessibility, user development support and strength in making 2D and 3D simulations (Arora, 2021). Apart from the gaming industry, Unity is also used by industries such as automotive, architecture, engineering, and construction (Juliani *et al.*, 2018; Unity Technologies, 2021). Hussain *et al.*'s (2020) survey showed that Unity has been playing an active role in the game development field, and the usage and industrial devotion to Unity is increasing and amplifying.

After reviewing the standards and tools for product modelling in sections 3.4 and 3.5, the following section explores some fundamental concepts of Design Engineering Automation with KBE to identify key aspects that should be considered when modelling a product using KBE system to support DEA.

3.6 Key Concepts of DEA with KBE System Development

For complex product design, the current engineering design process has shown an excessive imbalance between the time spent on non-creative activities and the time available for design innovation. It forces the KBE system to facilitate multi-fidelity, generative modelling of complex products and optimisation in a reliable and time-efficient manner (La Rocca and Tooren, 2012). The following sections provide a further review of different aspects that should be considered for Design Engineering Automation using KBE techniques.

3.6.1 Multi-Fidelity

Fidelity is the level of accuracy or complexity of a product model (Fernández-Godino *et al.*, 2016). A high-fidelity model (HFM) contains more details and has higher complexity of a product. As known, one common use of a product model is for simulations. Therefore, performing a simulation with an HFM is more time-consuming and computationally

expensive. In contrast, using a low-fidelity model (LFM) for simulations could save computation time, but the accuracy of the model itself may not meet the requirement.

In some simulation cases, surrogate models are built because the product data is too expensive to obtain or because there are regions where the data is not available. In other words, surrogate models are approximations that are fit to the available data of a phenomenon and make a functional relationship between input variables and the output quantities of interest. However, even for some high-fidelity models, performing simulations with fitted surrogate models may also be too expensive. To overcome this problem, the multi-fidelity model is utilised. It combines HFM and LFM to achieve proper accuracy at a reasonable cost. An HFM can be simplified to realise the multi-fidelity of a model. The simplification can be done in different ways, for example, by linearising the system, using averaged results in one dimension, using simpler physics models, less refined domain, or partially converged results (see Figure 3-6).

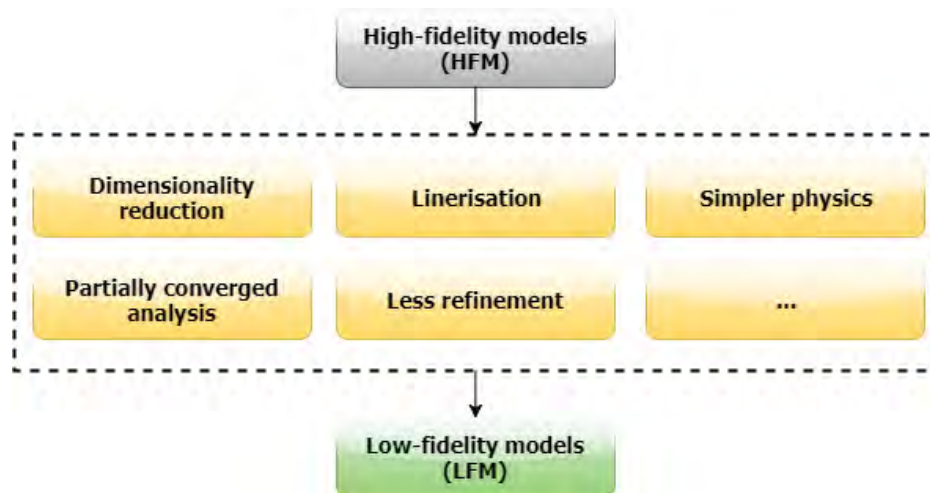


Figure 3-6: Converting HFM to LFM by different simplifications. Adapted from Fernández-Godino et al. (2016).

3.6.2 Generative Modelling

As a product usually inherits a number of structures from the previous design, when designing a new product, the lead-time required to determine a candidate model from alternatives is always a bottleneck. Therefore, if the product is modelled with the knowledge describing why and how the product is generated, the lead-time in design analysis and evaluation can be saved (Isaksson, 2003). A generative model is not just a CAD model but describes the engineering intent behind the geometric design. It captures the design strategy required to generate a particular product from a specification (La Rocca, Krakkers and van Tooren, 2002). In terms of generative modelling, the framework developed in this research can capture both geometrical information and its corresponding design knowledge. When designing a product variant in a generative modelling environment, the designer can search for a broader set of product design configurations rather than being restricted to simple parameter changes. A generative model provides designers with a set of possibilities based on the knowledge hierarchy from predefined products. This empowers designers to search for a broader set of product design configurations rather than being limited to simple parameter changes. For instance, pre-existing knowledge provides designers with the information required to analyse why the product is generated. This could also help designers to trace how the product is revised and updated. Consequently, the lead time will become shorter by using pre-existing knowledge defined in the product model.

3.6.3 Common Computational Model

According to Gilkinson et al. (2015), generative modelling cannot achieve complete integration of knowledge and design cycles by itself. To overcome this, common computational models (CCMs) can be employed to represent the design intent by storing the how, why and what of the design, connecting conceptual, preliminary and detailed models with information on the product and process, as well as the multi-fidelity requirements of

associated applications (Chapman and Pinfold, 2001). Furthermore, common computational models could allow seamless data exchange for all relevant engineering tools by providing a common interface and following an interoperable standard.

3.6.4 Design Optimisation

When a product is created to satisfy the required function, the designer tries to find the best solution for the task at hand. Hence engineering design optimisation (EDO) is performed, in which certain parameters need to be determined to achieve the best measurable performance under given constraints (Chang, 2016). And to perform EDO, knowledge about the design status, design variables, conditions, and relationships between the design variables are usually required (Roy, Hinduja and Teti, 2008). However, due to the lack of a detailed product knowledge base that predefines all the design variables and shows the interaction between different parameters and product performance, the optimisation process is often manual, time-consuming and includes a step-by-step method to identify the best combination of the product and driving parameters for the best solution. Thus, for saving time and manpower costs in optimisation, it is necessary to design products that can provide a full solution space based on existing design knowledge to automate the manual optimisation process.

After identifying key aspects that should be considered for developing a product model using KBE for DEA, the following section reviewed the recent relevant research works that have been done in KBE framework development and their implementation to support engineering activity.

3.6.5 Applying Key Concepts in KBE Product Modelling System Development

Based on the review from the above sections, to achieve product design in a shorter time with high quality and to support various applications of the product, the KBE system could take into consideration the concepts of multi-fidelity, generative modelling, common computational model and design optimisation (see Table 3-4). However, since this research

focuses on developing a method to enhance the product modelling process in product design, the product modelling system to be developed in this research does not require multi-fidelity for simulation. Nonetheless, the characteristic of multi-fidelity could still be achieved by varying the knowledge embedded in the product model.

Table 3-4: Characteristics of a product modelling system from the KBE perspective

Characteristics	Capability	Benefits	Reference
Multi-fidelity	Vary product complexity	Save time of simulations	(Fernández-Godino <i>et al.</i> , 2016)
Generative Modelling	Store design intent and product configurations information	Save lead-time in design analysis and evaluation	(La Rocca, Krakkers and van Tooren, 2002; Isaksson, 2003)
Common Computational Model	Provide a common interface to connect models with associated applications tools	Save model transferring time between different engineering tools	(Chapman and Pinfold, 2001; Gilkinson <i>et al.</i> , 2015)
Design Optimisation	Integrate rules to help identify the best combination of the product performance and driving parameters and avoid making mistakes in engineering tasks.	Achieve the best solution for engineering tasks	(Roy, Hinduja and Teti, 2008; Martins and Lambe, 2013; Chang, 2016)

3.7 Related Research Work in KBE Product Modelling Framework Development

Many frameworks and applications have been developed to support engineering automation through the application of KBE techniques in recent decades. Features and limitations of

KBE frameworks which are presented in the existing KBE methodologies (Lovett, Ingram and Bancroft, 2000; Melody Stokes, 2001; Chapman *et al.*, 2007; Rocca and Tooren, 2007; Curran *et al.*, 2010) have already been introduced and discussed in Section 3.2. This section discussed some of the existing frameworks and applications that addressed product modelling from the KBE perspective.

Hale (2002) introduced PACKS (The parametric composite knowledge system) and SCAD (The Steered Composite Analysis and Design System) as two knowledge-based software systems for composite design, analysis and manufacture. Even though these two systems are successful in the aerospace industry and other manufacturing sectors, they mainly specialise in composite structure design and fibre placement process. There is still a vacancy in the geometry representation of product modelling within KBE systems.

Adaptive Modelling language (AML) is utilised in both PACKS and SCAD systems to provide an object-oriented knowledge framework for representing products. It enables users to “construct a class hierarchy in which complex classes inherit properties from simpler classes” (Hale, 2002). However, as discussed in Section 3.5.2, AML software is not available to general users due to its limited accessibility.

Sanya and Shehab (2014) proposed an ontology framework for developing a platform-independent knowledge-based engineering system in the aerospace industry. It enables the reuse of knowledge and eliminates platform-specific approaches in the development of KBE systems. However, a limitation of their work is that the graphical display is limited to primitive shapes rather than a complex engineering product. The integrated knowledge base still needs improvement to capture the design rules of a complex product.

Many innovative KBE frameworks have been proposed to support manufacturing system interoperability (Giovannini *et al.*, 2012; Chungoora *et al.*, 2013; Fortineau, Paviot and Lamouri, 2013). However, these approaches are mainly focused on modelling manufacturing

engineering knowledge. Data transfer between CAD and KBE applications is mostly at the theoretical level (Schätzle, 2016). There is still a lack of KBE approaches that focus on modelling and transferring design engineering knowledge.

3.8 Literature Synthesis

3.8.1 Literature Review Summary

Chapters 2 and 3 conducted the literature review in the areas of Product Design, Design Engineering Automation, Product Modelling and Knowledge-Based Engineering. The literature review shows that embodiment design and detail design are the most time-consuming stages in the product design process and are most conducive to machine assistance. Computer aided tools and systems, such as CAD, CAE, CAM, and CIM, have shown great advantages in reducing the elapsed time, manpower, and resources expended in completing the embodiment design and detail design process for the creation and modification, analysis or optimisation of a product. However, CAD tools and systems require the designer to have knowledge and design experience of the product to judge the correctness of the function and understand what is going on beyond what is graphically shown on the computer screen. There is a need to integrate product design knowledge into a formal product model through the design process while using CAD.

In the literature, it is evident that design automation could enable companies and industries to deal with customised products more quickly and efficiently by automating repetitive design tasks in the existing product design processes. CAD models have been widely used as geometric product models for the automation of creating product variants in product modelling. The automation with CAD models is implemented through the parametric modelling method. However, the information stored in the CAD model is still limited to geometric data and parametric values. Extensive research has been carried out in the past to extend the CAD models by integrating CAD with CAE information. Two challenges in the

existing research are managing the product data in the complex products and the mismatch between the availability of information and the accessibility of the appropriate information to designers. There is a need to develop a product model which can capture and reuse complex product data and provide accessible and appropriate information to designers.

Therefore, this research will develop a product modelling framework that can capture and represent all required design knowledge in the product design process. The developed product model from the framework will serve as an accessible knowledge base that enables designers to work on the design tasks without previous design experience.

In the literature, various product modelling methods have been established to develop different product models. This research focuses on knowledge-based product modelling since it is characterised by capturing and reusing engineering knowledge such as human expertise and product and process knowledge in the modelling process.

Further, the literature review explored different knowledge-based engineering techniques for capturing and reusing knowledge. This research has chosen the knowledge capture methodology (KCM) as the knowledge capturing and reusing method, as this method has evolved from a natural understanding of the way design engineering work and presents clearly how parts and their relationships of a product are generated from a designer's perspective. The decomposition of product and its design knowledge has given KCM better capability with product modelling in terms of capturing, structuring and reusing design knowledge. Therefore, in this research, KCM will be used to capture and reuse the existing design knowledge in the product modelling process. However, one major shortcoming of the KBE techniques is the "black-box" problem. There is a lack of substantiation steps, as limited implementation advice and use case with tools and techniques examples are provided and discussed in the existing KBE methodologies. KBE systems are implemented in a way that is not readable and understandable to the end-users. Thus, there is a need to develop a

KBE implementation framework along with use cases and enabling tools for the purpose of capturing and reusing design knowledge in product modelling to support design engineering automation.

The conducted literature review further explored some key concepts in MBE and MBSE to identify how to represent product models within a neutral format and how to exchange product data through product models. This study will use UML/SysML to structure the product data, as they have been widely used as a platform-independent visual modelling language in various domains. In terms of product model data exchange, the STEP standard is selected as a neutral product modelling standard as it has been highly used plus supported in the industry and can provide steady data exchange among common CAD software. However, STEP does not support the exchange of parameters, design intent and other associated product model data. Although some research works have been done to enrich product model data in STEP through mapping and translation methods, problems such as data missing and conversion errors still exist. There is a lack of research on how to perform the integration of STEP and product modelling methodologies along with existing data resources. Thus, there is a need to develop a data exchange method that could not only enable product data exchange with STEP but also avoid the complex mapping process of the tedious STEP classes with the associated product model data (parameters, design intent, etc.) for product modelling.

Additionally, the conducted literature review explored existing product modelling and development tools. The existing CAD software does not support the application development and user interface development to apply KBE techniques, and AML is not available in this research due to the limited accessibility of its licence. Hence, Unity was selected as the most appropriate implementation tool for developing a product modelling environment using KBE techniques for this research investigation.

The conducted literature review further discussed different aspects that should be considered for Design Engineering Automation using KBE techniques. In this research, the product modelling environment that was developed through the use of KBE techniques needs to satisfy the following requirements from DEA perspective:

- 1) Generative modelling: store design intent and product configurations information
- 2) Common computational model: provide a common interface to connect models with associated applications tools
- 3) Design optimisation: integrate rules to help identify the best combination of the product performance and driving parameters and avoid making mistakes in engineering tasks.

Finally, the conducted literature review explored some relevant research work for the development of a product modelling framework using KBE techniques. The limitation of the existing relevant research work lies in the capture of design rules of the product model, and there is a lack of KBE approaches that focuses on modelling and transferring design engineering knowledge of a product in KBE applications.

3.8.2 Research Gap Summary

The outcome of the conducted literature review has identified research gaps that need to be addressed. A product model is seen as an information representation that provides data to build a product in a modelling process. The survey through existing product modelling methods, product model development and knowledge-based engineering techniques show that major issues in the product modelling process are:

- Existing CAD models do not provide enough design information in the product modelling process - lack of design knowledge representation of product model.

- Existing product models do not offer detailed implementation steps for the application. The interaction between the product model geometry and the design information remains unclear.
- Problem of availability and accessibility of design knowledge to designers - lack of knowledge capture and reuse in the product modelling process.
- “Black box” problem – lack of understanding and substantiation steps for the implementation of the KBE framework.
- Data transfer between CAD and KBE applications is mostly at the theoretical level - lack of KBE approaches for transferring design engineering knowledge of a product in KBE applications.
- Limited capability of knowledge capture and reuse in the existing CAD tools.

These identified research gaps from the literature have indicated the need for a knowledge-based product modelling framework to enable knowledge capture and reuse in the product modelling process.

3.8.3 Need for Knowledge Capture and Reuse in Product Modelling and Expected Contribution to Knowledge

The trend of product modelling for design engineering automation has evolved from manual drafting to CAD, from CAD to Computer Aided Product Modelling, and then to Knowledge-Based Product Modelling. The scope of product model development has also changed from single product design to product modelling for multiple product variants. The product data involved in product modelling has been expanded from “geometry-only” to “knowledge-integrated” as well.

As explained previously in Chapter 1, the higher demands of industrial development capacity, productivity and agile response to the market have forced the design engineering toward a shorter product development time. Therefore, less time is spent by experienced designers in

traditional mentorship and apprenticeship methods of design practice (Okudan and Zappe, 2006). The detailed literature review in chapters 2 and 3 has shown that the knowledge involved in the product modelling could not be formally collected and passed from experienced designers to new designers. Meanwhile, less-experienced designers are being given increasing and complex design tasks (Okudan and Medeiros, 2005). However, most of these less-experienced designers are not knowledgeable enough to undertake the design responsibilities independently due to a lack of engineering knowledge of the product. Moreover, the growing complexity of products has also driven the product development process to involve multidisciplinary teams. According to the American Society of Mechanical Engineers (ASME)'s article (Brown, 2020), during the new decade, design engineers have to collaborate with non-engineers to add embedded capabilities to optimise the design to meet other design requirements from different domains' aspects. However, the retirement of the last generation of design engineers adds more pressure on the less-experienced engineers in the industry. This problem must be addressed in the coming decade by the engineering profession.

In the literature, extensive research has been done to extend the current product models by integrating more product data with existing CAD models. However, the detailed application steps of these product models in the product modelling process are not provided in the literature, and the interaction between the product model geometry and the design information remains unclear. Different product modelling methods have been reviewed, and knowledge-based product modelling is identified as the most appropriate method for developing a knowledge integrated product model that can represent all required design knowledge to assist product modelling. Knowledge-based engineering techniques provide the capability of capturing and reusing knowledge in the product modelling process. However, the literature review of the existing KBE methodologies shows that there is a "black box"

problem in understanding KBE applications, and the substantiation steps for the implementation of the KBE framework are also limited.

The need for knowledge capture and reuse in product modelling is also addressed by Schätzle (2016). Data transfer between CAD and KBE applications is mostly at the theoretical level. Even though STEP has been widely used for data exchange between different CAD systems as a product modelling standard, it is still limited in providing parameters, design intent and other data for product modelling. Minimal research has been done in terms of how to perform the integration of a product modelling standard and product modelling methodologies along with existing design knowledge. There is a lack of KBE approaches for transferring design engineering knowledge of a product into KBE applications.

With regards to implementing KBE with existing CAD tools, the literature review has found that current CAD tools provide limited capability of enabling knowledge capture and reuse for product modelling process. This further addressed the need to provide a KBE implementation framework for knowledge capture and reuse in the product modelling process.

3.9 Chapter Summary

This chapter reviewed knowledge-based engineering methods and provided a detailed understanding of their utilisation for knowledge capture and reuse for product modelling. KCM is identified as the most suitable approach for capturing and reusing knowledge for product modelling. This chapter further discussed the key concepts that should be considered for applying KBE techniques for product modelling, which are an exchangeable data representation structure and a neutral standard. This chapter also reviewed the current product modelling standard and tools and provided the selection process of the most applicable product modelling standard and implementation tools for this research. Based on the conducted literature review, six significant research gaps were identified. The need for

knowledge capture and reuse in product modelling is discussed based on the findings from the literature, and the expected contribution to knowledge is presented. The next chapter extends the outcomes and formulates the actual methodology based on the findings from chapters 1-3.

4 Research Methodology

4.1 Introduction

This chapter first explains how the research questions and hypotheses are formed based on the findings from chapters 1-3 to provide an understanding of how the presented research methodology will address the identified research gaps. Also, this chapter includes a research plan to explain the journey of this study in a systematic and logical way. This research plan is further explained through four phases: literature review, research design, development, and evaluation. This chapter also provides the identified enabling methods for undertaking research development within the realm of reusing existing product knowledge into product modelling of engineering components. It highlights the applied methods in this research with justifications and explanations of why the selected methods best fit this research. This chapter also explains the chosen use case evaluation method and presents the evaluation criteria for this research.

4.2 Research Questions and Hypothesis Development

Chapter 3 has identified key research gaps that need to be addressed in this research. The presented research methodology will address these research gaps in relation to the research questions, objectives and hypotheses that have been established in this research. The research questions are:

- How can the design knowledge be structured and represented through a product model?
- How can this product model be implemented in a knowledge-based product modelling environment?

- How can the principles and practice of knowledge-based engineering be applied to capture and reuse the existing design knowledge for product modelling through a knowledge-based product modelling framework?
- How can this framework be implemented and applied by designers to enhance product modelling?

The following are the research objectives:

- To establish the research scope by identifying and reviewing the features and issues on product modelling for design engineering automation.
- To review methods, standards and tools used in product modelling for developing product models.
- To distinguish appropriate enabling methods and tools for capturing and reusing knowledge in product modelling.
- To develop methods of capturing, reusing, and exchanging design knowledge for product modelling and to develop a knowledge-based product modelling environment for applying these methods.
- To validate proposed methods of capturing, reusing and exchanging design knowledge in product modelling and evaluate the performance of the developed knowledge-based product modelling environment.
- To evaluate the modelling process within the developed knowledge-based product modelling environment, in contrast to the modelling process in existing CAD systems.
- To generate implementation guidance on how to use the approach for capturing and reusing existing product design knowledge to support design automation.

The research hypotheses are:

Research hypothesis 1: The proposed methods and tools for product modelling in KBE can improve the knowledge representation of product modelling by:

- Providing a generic knowledge integrated product model which can represent all associated product information, including geometric data from CAD and non-geometric information such as design intent, design parameters, design rules, etc.

Research hypothesis 2: The proposed methods and tools for product modelling in KBE can improve the existing KBE techniques for product modelling to support design engineering automation by:

- Formalising the product model in a neutral format and interoperable standard and generating a new data exchange method for transferring design engineering knowledge of a product in KBE applications.
- Enabling knowledge capture and reuse in the product modelling process through a knowledge-based product modelling framework,
- Developing a KBE application implementation framework for product modelling
- Providing detailed substantiation steps for the implementation through use cases and tools.

4.3 Research Plan

Before explaining the undertaken research methodology, it is important to outline a research plan to provide a systematic and logical overview of this research.

This research plan consisted of the following four research phases: literature review, research design, development and evaluation. The developed research plan is illustrated in Figure 4-1 and is further explained in the following Section 4.4.

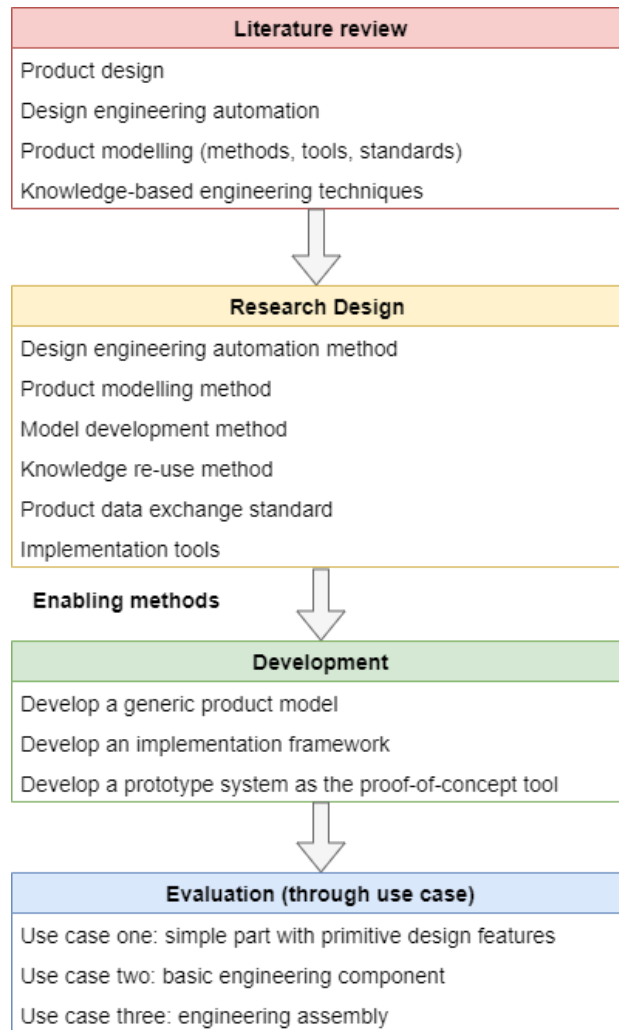


Figure 4-1: Research plan

4.4 Research Phases

To achieve the aim and objectives of this research and to conduct the research smoothly, this study has been split into four major phases (as shown in Figure 4-1). Phase one involved a detailed review and analysis of the literature that related to the key issues of this research. Phase two involved a research design that identified suitable enabling methods for this study. Phase three involved the design and development of the virtual product modelling framework and prototype system. Phase four involved the evaluation of the effectiveness of the developed method with three use cases.

4.4.1 Phase1: Literature Review

Literature review brings clarity and helps establish a fundamental comprehension of the current research methods and relevant work that have been done by other researchers on this research topic. In this research, the literature review started from understanding the product design process for product development to identify the key design stage that is most conducive to machine assistance. This led to further exploration of product design advancements with computer-aided technologies. Next, existing CAD models and methods used in design engineering automation were reviewed to assess strengths and weaknesses of integrating knowledge within the existing models. It helps to address the importance of product modelling in providing a complete product data representation to support the product design process. Later, different product modelling methods were studied and reviewed to identify the most appropriate product modelling method as state of the art. Furthermore, a detailed literature review was conducted to gain a deep understanding of the current knowledge-based engineering techniques for capturing and reusing knowledge, product modelling standards for product model development and tools for implementation. Advantages and disadvantages of these methods and tools were presented and relevant work in these areas were also reviewed and discussed. Finally, all of these reviewed studies, methods and relevant work were analysed and synthesised to help narrow down the research scope, formulate the research problems, identify the research gaps, address the need of this research, and establish the research design with enabling methods.

4.4.2 Phase 2: Research Design (Enabling Methods)

In order to address the research questions and fulfil the research objectives listed in Chapter 1, certain enabling methods have been identified and adopted from literature and related research work in these seven main topics: design engineering automation, product modelling, model development, knowledge re-use, product model data exchange and knowledge sharing

method and tools for implementation. A more detailed discussion of these enabling methods used in this research is presented as follows.

a) Design engineering automation

To support design engineering by implementing and reusing knowledge in solutions, tools or systems, Cederfeldt and Elgh (2005) pointed out that two aspects need to be considered in design automation, which are information handling and knowledge processing. In this research, information handling can be described as the reuse of CAD models, and knowledge processing refers to the reuse of existing knowledge, such as rules and constraints; both aspects are incorporated to develop the proposed framework.

b) Product modelling method

Product modelling has been regarded as a key technique to develop reusable CAD models to support design automation (Tay and Gu, 2002). The choice of a product modelling method to develop a generic model that enables the knowledge to be reused for design automation has been made by comparing the current product modelling methods from literature (Table 4-1). It can be seen that knowledge-based product modelling has the advantages of capturing and reusing the knowledge. However, limitations still exist in this method as the modelling method is always developed for a particular product in a platform-specific expert system. This could generate black-box problems when communicating between different systems (Cederfeldt, Elgh and Rask, 2006; Fan and Bermell-Garcia, 2008). To overcome these limitations, this research developed a generic product model that can serve as an adaptable, structured and reusable knowledge base for different platforms.

Table 4-1: Comparison of product modelling methods

Product Modelling Methods	Characteristic	Advantage	Limitation
Solid Product Modelling	<ul style="list-style-type: none"> • Use mathematical principles 	<ul style="list-style-type: none"> • Mature tools in modelling three-dimensional objects 	<ul style="list-style-type: none"> • Only geometric information
Feature-based Product Modelling	<ul style="list-style-type: none"> • Extension of solid product modelling. • Relate design intents and functionality with geometry 	<ul style="list-style-type: none"> • Mature tools in modelling high-level shapes 	<ul style="list-style-type: none"> • Not able to transfer knowledge such as expertise and experience to other designers
Knowledge-based Product Modelling	<ul style="list-style-type: none"> • Reusing engineering knowledge in the modelling process. 	<ul style="list-style-type: none"> • Reduce unnecessary re-analysis, re-design, and re-planning. • Simplify the modelling tasks and ensure the modelling quality 	<ul style="list-style-type: none"> • Product and platform-specific • Black-box problems

c) Model development method

A range of product models and systems were studied in the literature to understand what the general methods are to develop a model. A list of comparisons is shown in Table 4-2. It can be seen that to develop and represent a product model with the existing knowledge, key elements need to be considered and involved, such as relationship, logical frame or structure, model class, rationale, knowledge-reuse, etc. These key elements were identified and used in this research to develop a product model which could represent the product with existing knowledge.

Table 4-2: Key elements in related research work from literature

Model or system developed in literature	Adopted methods	Key elements	Reference
Featured-based and rule-based knowledge representation	Represent knowledge using feature	Feature of the product	(Jurit H., Saia, A. and De Pennington, 1990)
Frame-rule structure for mould product design system	Represent knowledge using numbers of frames related to each other by relationship	Relationship between parts and parameters	(Lou, Jiang and Ruan, 2004)
Axiomatic information Model for Design (Aim-D)	Formal basis and logic of product structure	Logical product structure	(Salustri, 1996)
Core Product Model (CPM)	Non-geometric information class	Non-geometric class	(Fenves, 2001)
Product Family Evolution Model (PFEM)	Extend CPM by adding rationale of the changes	Design intent and rational	(Wang <i>et al.</i> , 2003)
Open Assembly Model (OAM)	Extend CPM with assembly relations	Assembly relationship	(Mehmet <i>et al.</i> , 2005)
A unified central product model in UML	Extend CPM with assembly relations	Assembly relationship	(Gross <i>et al.</i> , 2009)
A simulating UML model of the FireSat mission satellite	UML classes with CAD model	UML diagram	(Gross and Rudolph, 2012)
Multi Model Generator for Aircraft Design	UML KBE Techniques	Knowledge representation	(Rocca, 2011)

d) Knowledge re-use method

From the literature review and analysis, Knowledge-Based Engineering (KBE) has been identified as an appropriate method that is applicable in developing the proposed virtual product modelling framework for the benefits of capturing and reusing engineering knowledge in the product modelling process. A range of KBE approaches has been developed in the literature with the aim of reducing the time and costs of product developments by automating the repetitive design tasks and optimising the design process in all aspects of the design process. These methods, such as MOKA (Melody Stokes, 2001), KOMPRESSA

(Lovett, Ingram and Bancroft, 2000), KNOMAD (Curran *et al.*, 2010) and KCM (Chapman *et al.*, 2007), have been successfully used in different engineering areas such as automation, civil engineering and aerospace engineering in terms of modelling and cost-saving (Rosenfeld, 1995). In this research, to overcome the deficiency of knowledge reuse and availability for the end-user, KCM (Knowledge Capture Methodology) has been adopted as it is suitable for parametric geometry representation where components are associated with parametric values in the KCM process (Terpenney, Strong and Wang, 2000). The application of KCM makes the dimension of the product constrained by the parameter values from the existing knowledge, which help to maintain the design intent when design modifications are performed. It also enables a flexible modelling process as the product model can be modified by changing parameters for a quick generation of product variants.

e) Product model data exchange and knowledge sharing method

An international standard is followed in this research to improve the interoperability of the product model among CAD software and avoid data conversion faults. By comparing three international standardised formats (see Table 4-3), STEP has been identified as an appropriate standard since it has been widely used in most the product modelling tools and provides a standardised, comprehensive information representation for different engineering application tools.

Table 4-3: Comparison between IGES, STEP and JT

Standard	Usage	Advantage	Limitation
IGES	CAD Data Exchange	-	Decreasing usage
STEP - 242	CAD/PLM Data Exchange	Product Manufacturing Information (PMI)	Under development. Rarely supported by current CAD vendor.
STEP - 203	CAD/PLM Data Exchange	High usage and wide dissemination	Only non-geometric. Tedious classes.
JT	Data Exchange	Fast Visualisation	Binary Format. Not readable.

			Not widely used.
--	--	--	------------------

A new STEP Application Protocol AP242 (*STEP AP242 Project*, 2014) was released in 2014, which was claimed to cover Product Data Management, 3D model-based design with Product Manufacturing Information (PMI), etc. However, this new protocol is still under development and is rarely supported by the current CAD vendors (Schätzle, 2016). An obvious drawback of the STEP format is it does not allow for the exchange of parameters, design intent and other data that may be associated with the CAD models. Moreover, the readability of the STEP file is low for the end-users. Some research work has been done to enrich the product model data in STEP by mapping data into STEP (Arnold and Podehl, 1999; Kahn *et al.*, 2001; Barbau *et al.*, 2012). However, mapping with the entire STEP standard is complicated, time-consuming and results in data missing (Barbau *et al.*, 2012). Only mapping extra data with STEP does not help improve the re-usability of the knowledge because non-geometric information is simply stored as context and not linked with geometry. To address these limitations of STEP and avoid the complex and tedious mapping process, this research provides a novel way of integrating product non-geometric information into the product by using a knowledge file. STEP is only used as a format to store geometric data for the representation of the shape of the product. All the key parameters, design intent and other non-geometric data were classified and stored in a knowledge file in Extensible Markup Language (XML) format, which could be used as an independent interoperable format to facilitate the store, exchange and re-use of knowledge. This data exchange method will be further explained in Chapter 5.

f) Tools for implementation

In order to implement the developed methodology, this research has reviewed current CAD tools (see Table 4-4) to identify the best tools that could be used to visualise the developed product model and develop a prototype system that could work as a proof-of-concept tool.

Table 4-4: Reviewed implementation tools (version up to 2021) in this research

Tools	Open Source	Geometry View	Structure View	Application Development	Interface Design	Data Capture and Reuse	Availability
AML	x	✓	✓	x	✓	Only in proprietary format	License required
SolidWorks	x	✓	✓	x	x	Only in proprietary format	License required
CATIA	x	✓	✓	x	x	Only in proprietary format	License required
Siemens NX	x	✓	✓	x	x	Only in proprietary format	License required
Autodesk Inventor	x	✓	✓	x	x	Only in proprietary format	License required
Creo	x	✓	✓	x	x	x	Free
CAD Exchanger	x	✓	✓	x	x	x	Free
STP viewer	x	✓	✓	x	x	x	Free
IDA-STEP	x	✓	✓	x	x	x	Free
CAD Assistant	x	✓	✓	x	x	x	Free
Free CAD	✓	✓	x	x	x	Only in proprietary format	Free
Unity	✓	✓	✓	✓	✓	Through Application Development	Free
Unreal	✓	✓	✓	✓	✓	Through Application Development	Free

Finally, the developed methodology was applied to a KBE product modelling environment developed in Unity 3D software. Unity 3D has a license-free version for personal development, which provides an environment for the development of interactive 2D and 3D content, including a rendering and physics engine and a scripting interface to program interactive content (Unity Technologies, 2020). It allows the developers to export their applications into all the mainstream operation systems (Windows, Mac, Linux, IOS, Android) through multiple platforms (desktop, web, mobile).

Table 4-5 shows a summary of solutions with enabling methods for solving the identified problems from the literature. These enabling methods were further applied in Phase 3 for the development of the proposed virtual product modelling framework.

Table 4-5: Summary of solutions for solving the identified problems

Identified problems/research gaps	Solution with enabling methods
Need for design engineering automation	Information handling: re-use of CAD models Knowledge processing: re-use of existing knowledge
Only geometry information and limited design intent could be transferred by the traditional product modelling method. Existing CAD models do not provide enough design information in the product modelling process - lack of design knowledge representation of product model.	Knowledge-based product modelling method to develop a generative product model with knowledge classes
Problem of availability and accessibility of design knowledge to designers - lack of knowledge capture and reuse in the product modelling process.	KBE technique - Knowledge Capture Methodology
Existing product models do not offer detailed implementation steps for the application. The interaction between the product model geometry and the design information remains unclear.	Develop a generic and platform-independent product modelling framework that enables knowledge capture and reuse: <ul style="list-style-type: none"> • To develop a generic model: key elements need to be involved as knowledge • To drive the geometry by reuse of design
“Black box” problem – lack of understanding	

and substantiation steps for the implementation of the KBE framework.	<p>rules</p> <ul style="list-style-type: none"> • To enable cross-platform data exchange: STEP and XML • Provide detailed instantiation steps for applying the framework with use cases and enabling tools
<p>Limitation of exchanging parameters, design intent and other data in STEP. Low readability of STEP file. Mapping with STEP is complicated and results in data missing</p> <p>Data transfer between CAD and KBE applications is mostly at the theoretical level - lack of KBE approaches for transferring design engineering knowledge of a product in KBE applications,</p>	Data exchange method - Only geometry data of the product model is stored in the STEP file. Other knowledge would be stored in a separate knowledge file.
<p>Current implantation tools do not support application development and interface design or are not license-free.</p> <p>Limited capability of knowledge capture and reuse in the existing CAD tools.</p>	Unity 3D and a knowledge capture tool

4.4.3 Phase 3: Development

The research design provides an in-depth understanding of the research issues and solutions to address the research gaps in capturing and reusing knowledge for product modelling. The enabling methods identified in Phase 2 are utilised in the development phase for implementing the proposed solutions, as shown in Table 4-5. This phase aims to design and develop a generic virtual product modelling framework that enables product modelling with the utilisation of the integrated existing product design knowledge in it. The development phase includes the following three stages:

- Development of a generic product model,
- Development of an implementation framework,
- Development of a knowledge-based product modelling environment as the proof-of-concept tool (prototype system)

During the development phase, a knowledge-based product modelling environment that enables capturing and reuse of the existing knowledge in product modelling is developed. Figure 4-2 shows the implementation framework developed for the conduction of research development. The development phase is discussed further in Chapter 5.

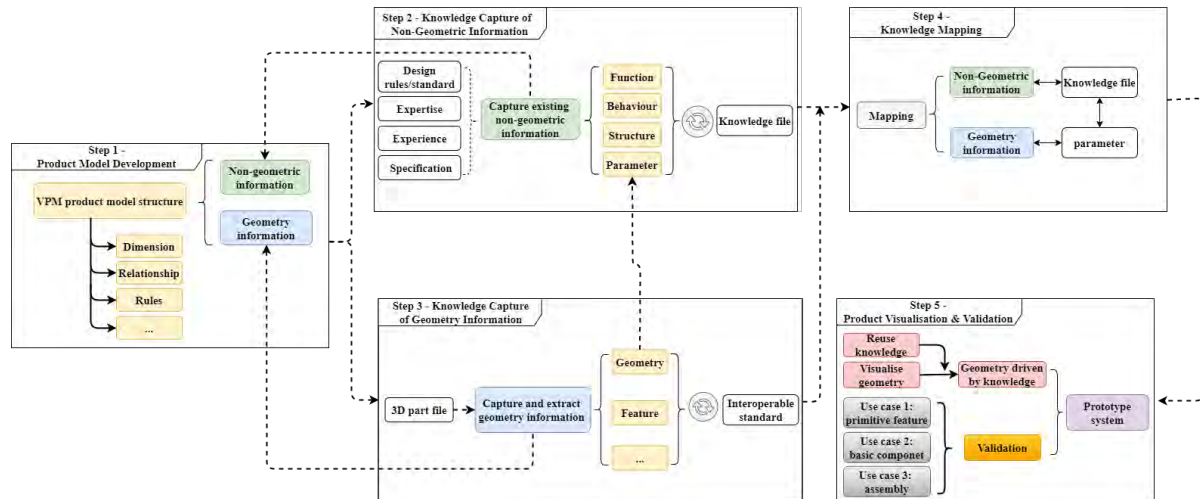


Figure 4-2: Implementation framework for the development

4.4.4 Phase 4: Evaluation with Use Cases

The main purpose of this evaluation phase is to assess and validate the effectiveness of the developed product model, implementation framework and proposed knowledge-based product modelling prototype system. The selection of a design evaluation approach is not unique but depends on the designed artefact and selected evaluation metrics. There are various evaluation approaches, including observational, analytical, experimental, testing and descriptive (Alan Hevner, Jinsoo Park, 2004). Table 4-6 shows the summary of the available evaluation methods identified from the literature.

Table 4-6: Design Evaluation Methods. Adapted from Hevner *et al.* (2004).

1. Observational	Case Study: Study artefact ¹ in depth in business environment
	Field Study: Monitor use of artefact in multiple projects

¹ Artefact can be described as the product produced during the development process. In this research, artefact refers to the product model, framework and prototype system.

2. Analytical	Static Analysis: Examine the structure of artefact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artefact into technical IS architecture
	Optimisation: Demonstrate inherent optimal properties of artefact or provide optimality bounds on artefact behaviour
	Dynamic Analysis: Study artefact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artefact in controlled environment for qualities (e.g., usability)
	Simulation – Execute artefact with artificial data
4. Testing	Functional (Black Box) Testing: Execute artefact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artefact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artefact's utility
	Scenarios: Construct detailed scenarios around the artefact to demonstrate its utility

Since this research is proposing an innovative virtual product modelling framework that enables the reusability of knowledge in the product modelling process, there is currently no such ready-to-use software or system to validate this work. To overcome this shortage of tools, a knowledge-based product modelling environment was developed to apply this framework and examine whether it will work in the proposed way. This validation requires controlled experiments, simulation, functional testing, and structural testing. Therefore, experimental and testing have been chosen as the evaluation methods in this research to validate the proposed model and framework.

Experimental and testing use cases are currently widely adopted to evaluate various design products such as virtual models (Isaksson, 2003; Cho *et al.*, 2016), prototypes (Van Holland and Bronsvort, 2000; Yoshioka, 2001; van Tooren *et al.*, 2005; Martínez-Pellitero *et al.*, 2011), scenarios (Haynes and Skattebo, 2004), systems (Chen and Wei, 1997; Lagos, 2007; Al-ashaab *et al.*, 2012) and interface (Bonnie E. John and Mashyna, 1997). In this research, three use cases were selected according to the Use Case Evaluation (UCE) guideline

(Hornbæk *et al.*, 2007) and employed to test the workability and effectiveness of the framework and the usability of the developed prototype system. The evaluation examines important aspects, including testing of generic representation of common engineering products, workability of system and re-usability of the existing knowledge that is integrated into the developed model in the product modelling process. The existing information collected for identified knowledge source of each use case was utilised for the evaluation of the proposed virtual product modelling framework.

Design work is complete and effective when it meets the requirements and constraints of the problems it was meant to solve (Hevner *et al.*, 2004). In this research, a Virtual Product Modelling Framework (VPM) has been developed that enables the existing knowledge to be captured and reused in the modelling process by applying the selected enabling methods to solve the identified problems. Hence, based on the research aims and objectives, identified research gaps and proposed solutions, the following criteria are derived and used to evaluate the workability and effectiveness of this research work:

- The capability of generative representation of engineering product in VPM
- The capability of the VPM to capture the product geometry and its associated knowledge from the existing product information
- The capability of the VPM to visualise the product geometry and its associated knowledge against use case data
- The capability of the VPM to present every part of the product and the relationships among them against use case data
- The capability of the VPM to propagate changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge
- The correctness of the changes applied to the product geometry by reuse of the existing knowledge against use case data

- The capability and correctness of the product geometry data exchange between different platforms against use case data
- The capability and correctness of the knowledge exchange through knowledge file against use case data

The evaluation process was further explained in Chapter 6. As a result of the evaluation, the proposed research framework was validated, and limitations of this research were identified for future improvement.

4.5 Chapter Summary

This chapter outlined the research plan and provided a detailed explanation of how the research was conducted through the research phases. This chapter also compared and identified suitable methods to establish the research design from seven major topics: design engineering automation, product modelling, model development, knowledge re-use, product model data exchange and knowledge sharing method and tools. This research adopted testing use case as the validation method for evaluation. The research design, research methodology and the proposed implementation framework were established in this chapter. The next chapter explains in detail the development of the proposed framework.

5 Virtual Product Modelling Framework

This chapter focuses on the explanation of the virtual product modelling framework for design engineering automation. It shows the development process of this virtual product modelling framework and explains how it can be further implemented with these identified enabling methods. Further, it provides detailed implementation steps of this virtual product modelling framework. Finally, a knowledge-based product modelling environment is presented, which is developed based on the implementation framework.

5.1 Virtual Product Modelling Framework Development

A virtual product modelling framework for developing a knowledge-based product modelling environment that enables existing knowledge to be captured and reused in product modelling has been developed in this research, as shown in Figure 5-1. This framework consists of five stages, namely:

1. Product model development
2. Knowledge capture of non-geometric information
3. Knowledge capture of geometry information
4. Knowledge mapping
5. Product visualisation and validation

In the first stage, a generic product model structure that can represent all required design information of the product will be developed using the meta class of VPM. Further, non-geometric information and geometry information of the product will be captured as existing knowledge in the second and third stages. Then, the interaction between the non-geometric information and geometry will be built in the knowledge mapping stage to provide the knowledge reasoning for the product modelling process. At last, the developed product model from VPM will be visualised and validated within a knowledge-based product modelling

environment through different testing use cases. Detailed explanations of these five stages are provided in the later sections.

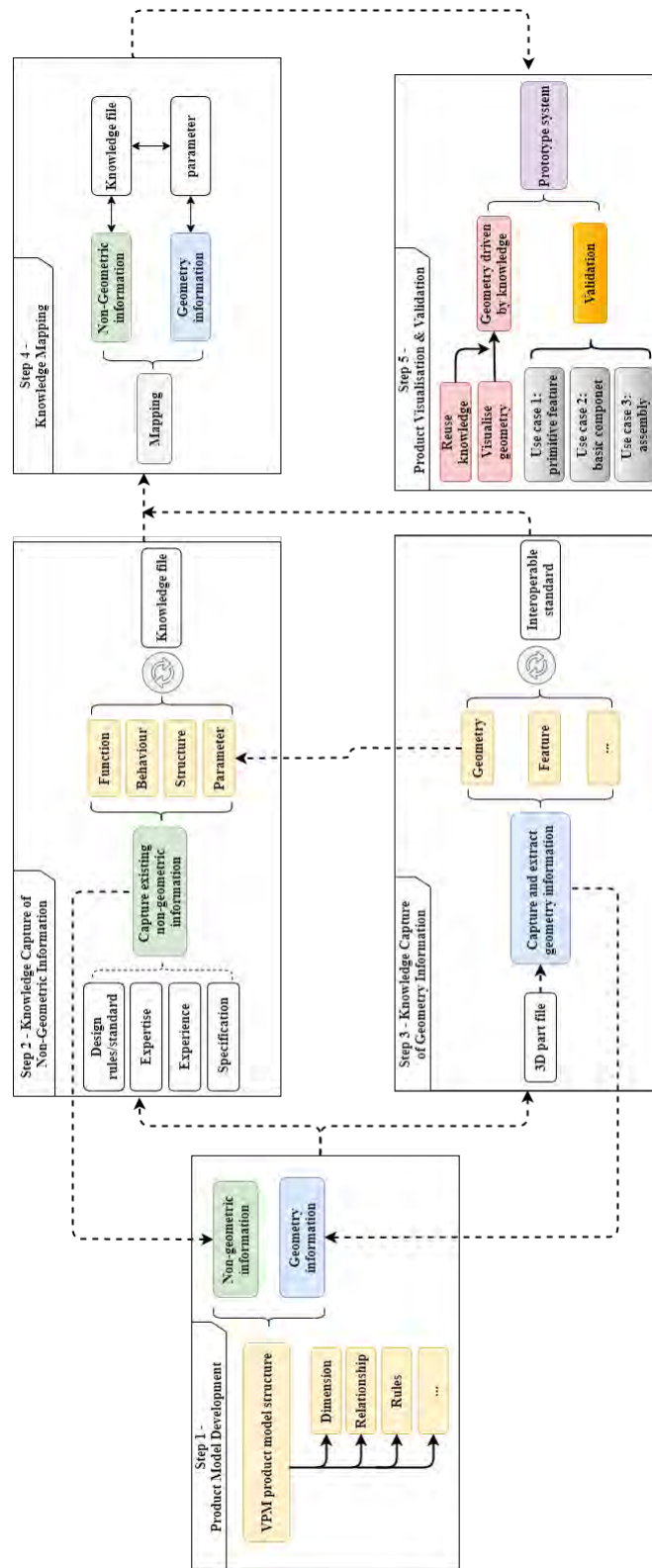


Figure 5-1: Virtual product modelling framework

This framework provides a set of activities for designer engineers to conduct for implementing the knowledge-based engineering techniques in the modelling process. As previously discussed in Chapter 3, KCM has been identified as the most appropriate method for capturing product design knowledge in the product modelling process as it shows the better capability of decomposing a product model and its design knowledge for knowledge capturing and structuring. It also provides the ability for parametric geometry representation as components are associated with parametric values after the KCM process. The eleven steps of implementing KCM are (Terpenny, Strong and Wang, 2000):

1. Select a product or process to model.
2. Decompose the product into atomic components.
3. Assign attributes to atomic components.
4. Establish atomic instances in database or tabular data source.
5. Create component classifications for the atomic instances.
6. Create use cases by grouping atomic components into assemblies.
7. Introduce a use case for each existing component classification.
8. Define a set of relations and create relationships between considerations.
9. Use relationships between components to propagate parametric values and apply selection constraints.
10. Associate parametric values of components for reporting and visualisation.
11. Repeat the process.

In order to deploy KCM in this research for capturing and reusing knowledge, each VPM framework step is required to map with the above eleven KCM steps. Table 5-1 shows how the developed VPM framework is mapped with KCM steps. Each step of the VPM framework is further explained in the following sections.

Table 5-1: Mapping between KCM steps and the VPM framework

KCM steps	VPM framework steps	Outcome
1,2,3,5	Product model Development	VPM product model structure - UML diagram
3,4,5	Knowledge capture of non-geometric information	Knowledge file - generated from the developed knowledge capture tool
3,4,5	Knowledge capture of geometric information	STEP file - exported from CAD
6,7,8,9	Knowledge mapping	KBE product modelling tool developed in Unity
10,11	Product visualisation & validation	KBE product modelling tool developed in Unity

5.1.1 Product Model Development

The product model structure (as shown in Figure 5-2) is a comprehensive and generative representation of the product with all its essential non-geometric and geometric information. It provides a data structure that includes all blocks describing a product from different aspects. Moreover, this hierarchical product structure also possesses inheritance characteristics as it is developed with the object-oriented concepts in UML. VPM meta classes are defined to constitute the product model, and these VPM classes are further explained in Section 5.2. A generic product model will be formed by integrating the knowledge captured in the second and third stages into this product model structure.

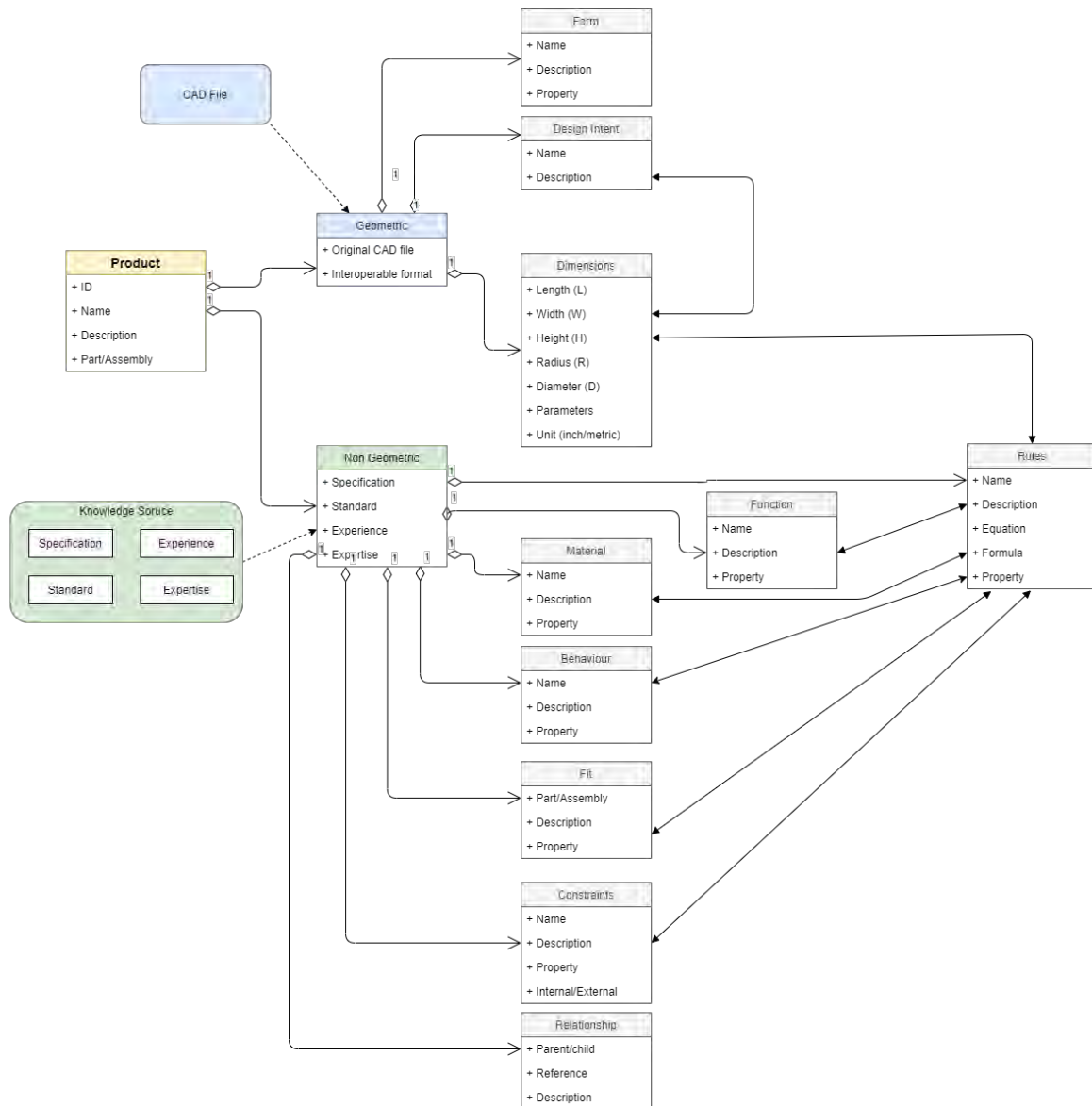


Figure 5-2: Developed product model structure from VPM

5.1.2 Knowledge Capture of Non-Geometric Information

As discussed in the previous chapters, in this research, knowledge is classified into two categories, which are non-geometric and geometry. The second stage of the methodology framework is to capture the existing non-geometric knowledge of the product. It includes identifying essential aspects that should be considered in the product modelling process, defining meta classes for each non-geometric aspect, capturing the existing non-geometric information, and breaking the non-geometric information down into the classified components.

Tasks that need to be performed in this stage are listed below (also shown in Figure 5-3):

1. Identify the essential non-geometric information used to specify the product from the existing design knowledge.
2. Decompose the non-geometric information into atomic classes.
3. Identify the rules that are applied in the design process of the product. Then the design engineer can then model the interactions between the non-geometric information and geometry.
4. Export the non-geometric information into a generalised format that can be reused and transferred between different industry APIs.

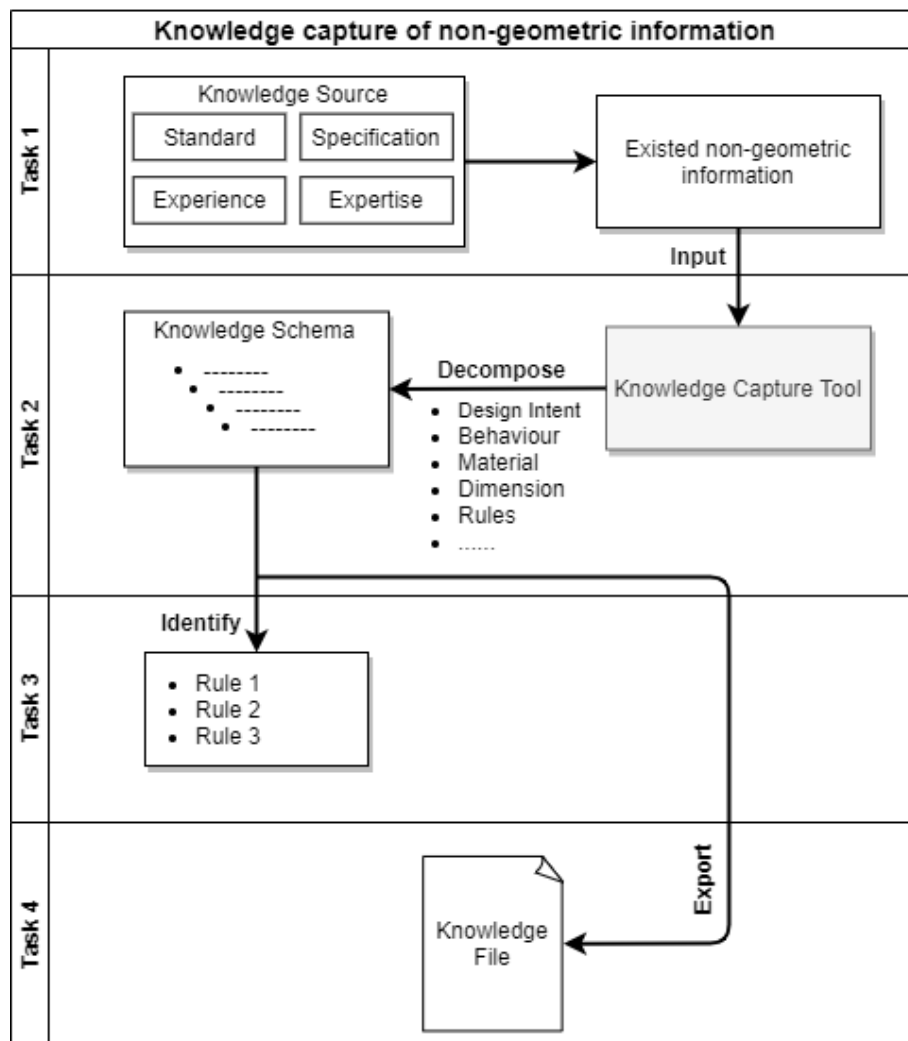


Figure 5-3: Task flow of capturing the non-geometric information

After performing these tasks, a knowledge file containing well-decomposed non-geometric classes and properties is generated. This knowledge file represents the product from non-geometric aspects and can be used as a knowledge base to share the product's information between design engineers. It can be further utilised to enhance product modelling by visualising the knowledge in a knowledge-based product modelling environment.

5.1.3 Knowledge Capture of Geometric Information

The next step in the virtual product modelling framework is to capture the geometric information of the product. In Design Engineering Automation, the geometric information of a product model can be automatically saved into a digital part file in the current CAD platform/software. An international standard needs to be followed as an interoperable format to allow geometric data exchange among different CAD platforms/software and to avoid data conversion faults. As discussed in Chapter 4, the STEP standard has been identified and used in this research as an interoperable format for exchanging product geometric information among different CAD platforms/software. Tasks performed in this stage are shown below:

1. Model the product geometry in the CAD platform/software.
2. Export the product CAD model into an interoperable and standardised format.

The result of performing these tasks is the geometry file with the geometric data that describes the product's geometry and can be used to exchange among different CAD platforms/software. Figure 5-4 describes the task flow of this stage, and the knowledge capture process is further discussed in Section 5.3.2.

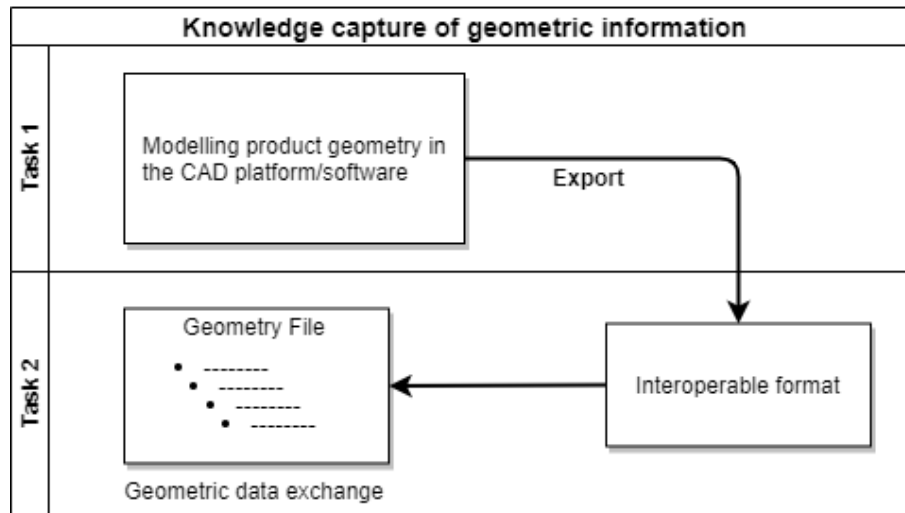


Figure 5-4: Task flow of capturing the geometric information

5.1.4 Knowledge Mapping

Knowledge Mapping is the most significant stage of the framework, which build the connection and interaction between the non-geometric information and geometry. In this stage, non-geometric information is transferred from the knowledge file to an object-oriented programming environment, where key parameters and design rules are linked to the product geometry. The data from the knowledge capture tool is exported into a knowledge file in an XML format and parsed using object-oriented programming. XML has been widely used as a generalised data format that is both human-readable and machine-readable for data exchange between different Application Programming Interfaces (APIs) in the industry. As identified in literature, knowledge reasoning can be achieved through the use of rules through object-oriented techniques. Since various knowledge classes and properties have been defined in the proposed virtual product modelling framework, object-oriented programming is employed as a programming paradigm that provides high modularity and reusability for these knowledge classes and properties. Table 5-2 shows how the rules are implemented in this research to provide knowledge reasoning in the knowledge-based product modelling environment.

Table 5-2: Illustration of implementation method of rules in this research

Rules	Description	Implementation method
Logic rules	IF-THEN-ELSE rules and conditional rules	IF-THEN-ELSE statement in object-oriented programming in different testing scenarios
Math rules	Mathematical rules, including trigonometric functions and operators for matrices and vectors algebra	Mathematical rules in object-oriented programming in different testing scenarios
Geometry handling rules	The generation and manipulation of geometric entities and parametric rules	Function of making change to dimensions in the interface
Configuration selection rules	Combination of mathematical and logical rules to change and control the topology of the product model.	Propagate the change of dimensions with knowledge reasoning text in the interface
Communication rules	Specific rules that allow data communication and interaction with other applications.	Adopting interoperable standards and formats for data exchange

Further, a knowledge mapping framework has been developed (as shown in Figure 5-3) to explain how the knowledge reasoning is performed in this knowledge mapping stage.

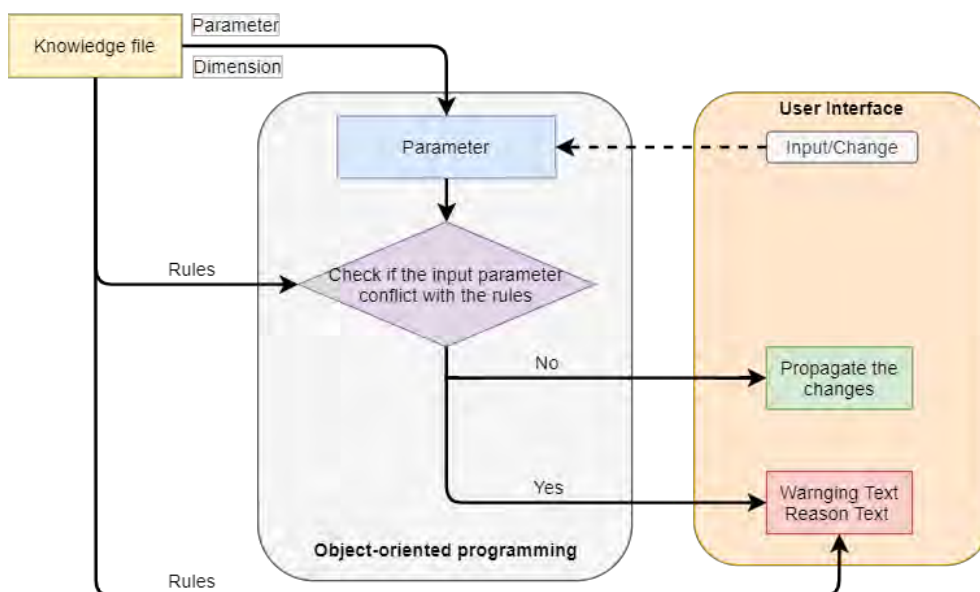


Figure 5-5: Knowledge mapping framework for knowledge reasoning

5.1.5 Product Visualisation and Validation

The last step is the visualisation and validation stage, where the developed product model in VPM is visualised and checked for the correct representation of the initial model. As discussed in Section 4.4.4, testing use cases are selected to identify and evaluate the proposed framework's effectiveness. Suppose the design engineer changes one dimension of the geometry of the initial product model; in that case, the virtual product modelling framework will check the rules that determine this geometry and propagates the rules and changes to the design engineer. In this case, design engineers will know what will be affected if the geometry is changed in this model and the constraints of these changes.

Visualisation is an integral part of the proposed framework as it provides a straightforward and effective way of understanding the product model. The developed framework was applied to a KBE product modelling prototype system developed in Unity 3D software. Unity 3D has a license-free version for personal development, which includes an environment for the development of interactive 2D and 3D content, a rendering and physics engine and a scripting interface to program interactive content (Unity Technologies, 2020). It allows the users to export their applications into all the mainstream operation systems (Windows, Mac, Linux, IOS, Android) through multiple platforms (desktop, web, mobile).

5.2 Product Model Development in VPM

As discussed in Chapter 4, to develop a generic product model that can represent all required design information as a knowledge base that provides guidance for design engineers to work on the design tasks without previous design experience, two key aspects need to be considered which are: re-use of CAD model and re-use of existing knowledge. To create such a generic product model, the following steps are followed in this research:

- To define meta class as the building block of the product model
- To apply an international standard and interoperable format so that the product model can be exported and exchanged among different product modelling tools

The defined meta classes will be used to develop the structure of the product model, and the existing design knowledge will be captured and decomposed as entities for these meta classes. The following sections provide further explanation of the VPM meta class and show how data represented by these meta classes are exchanged in this research.

5.2.1 Meta Class of VPM

a) Defining a product

From the literature, a product is defined as a physically realisable object that is produced by a process. As this research focuses on product modelling for Design Engineering Automation, the term product is defined as an engineering component produced by a process throughout this thesis. A child product within a parent product is termed as a “part”, and the parent product is termed as an “assembly”. In this research, a product model can be a representation of either a single engineering component or an assembly that consists of different engineering parts. Figure 5-6 below shows the relationship between an assembly product model and other parts of the product model.

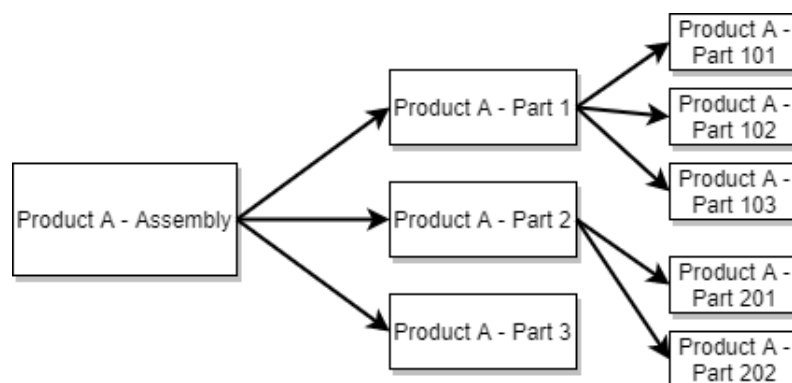


Figure 5-6: Example of the relationship between the assembly and parts of the product model (named “Product A”)

b) Defining meta class

Since the current approach is used for product modelling for Design Engineering Automation, extra supporting data need to be integrated to represent the product model. The meta class sets defined for the present research (as shown in Figure 5-7) are derived from literature review analysis and previous related research in product model development.

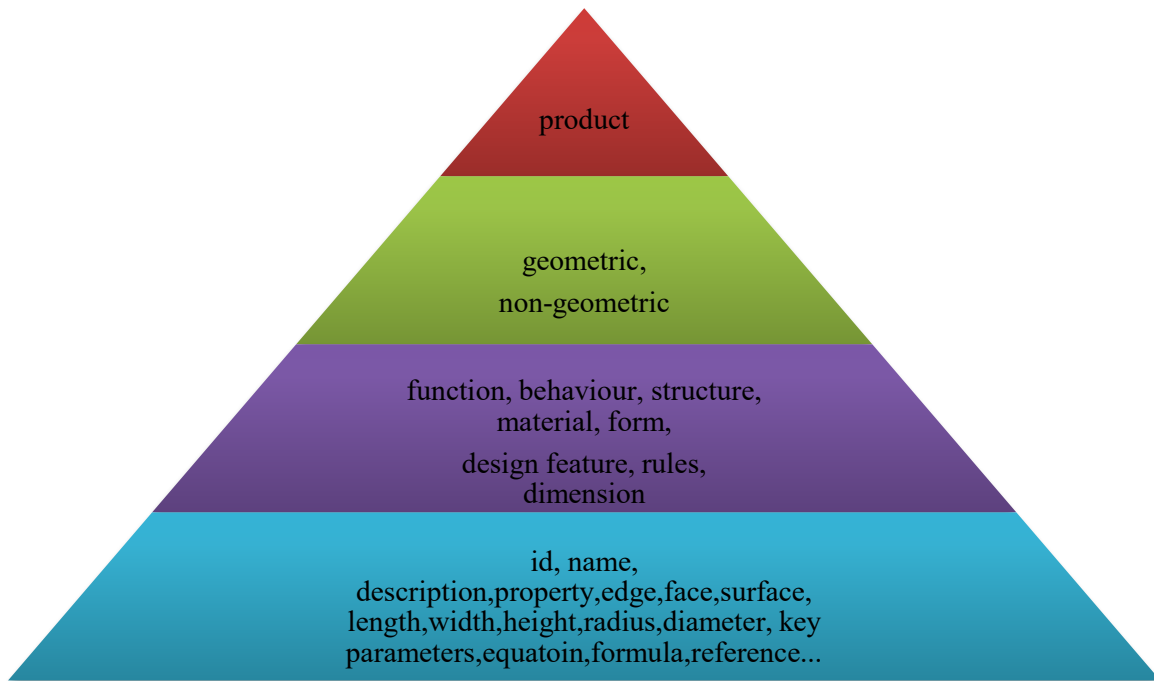


Figure 5-7: Product model structure data from literature. Adapted from Fenves *et al.*, (2004), Siemens PLM (2019b) and Boy *et al.*, (2015).

The lower the pyramid, the more detailed product data it provides. In this research, a product (top level of the pyramid) consists of two kinds of data which are geometric and non-geometric (second level of the pyramid). The geometric data describes the product's geometry using a standardised format that is exported from CAD. The non-geometric data is captured from the existing design knowledge to describe the product from different aspects of the product. The captured non-geometric data will be decomposed into the meta classes in the third level of the pyramid to provide a generic representation of a product. The bottom level of the pyramids provides detailed entities for the meta classes. In this thesis, a product model

can be represented the following meta classes: Function, Behaviour, Form, Material, Design Intention, Dimension, Rules, Fit, Constraint, Relationship. The “Structure” will be shown through the composition of the developed product model structure itself.

c) Geometric classes for STEP entity

After defining the meta class of VPM, the next step is to determine the geometric classes for the geometric data from STEP. As identified in the previous literature review, STEP has been used as a steady exchange format and dependable application interface between different computer systems and CAD software. Therefore, the geometric data of the product model can be automatically stored into pre-defined STEP entities when exporting the CAD model into STEP format. In this research, the geometric data is stored and exchanged through the STEP file itself. The definition of geometric classes for STEP entities is not required. However, it is still key to understand how the external geometric class can be mapped with the STEP entity in the current CAD software. Table 5-3 shows an example of how the geometric class could be defined and mapped with the STEP entity in a commercialised CAD software used in the industry.

Table 5-3: External geometric classes mapping with STEP Entity. Adapted from (Siemens PLM, 2019b)

Geometric class	STEP entity
Solid body	MANIFOLD SOLID BREP
Solid body with voids	BREP_WITH_VOIDS
Sheet body	MANIFOLD_SURFACE_SHAPE_REPRESENTATION →SHELL_BASED_SURFACE_MODEL
Wire body	GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION N→GEOMETRIC_CURVE_SET
Shell	CLOSED_SHELL (for solid bodies) ORIENTED_CLOSED_SHELL (for B-rep with voids, the outer shell is a ORIENTED_CLOSED_SHELL)

	OPEN SHELL (for sheet bodies)
Face	ADVANCED FACE
B-surface	B SPLINE SURFACE
Cone	CONICAL SURFACE
Cylinder	CYLINDRICAL SURFACE
Plane	PLANE
Sphere	SPHERICAL SURFACE
Torus	TOROIDAL SURFACE
Revolved surface	B SPLINE SURFACE
Extruded surface	B SPLINE SURFACE
Offset surface	B SPLINE SURFACE
Blending surface	B SPLINE SURFACE
Degenerate (apple/lemon)	B_SPLINE_SURFACE
Torus	
Loop	FACE_BOUND→EDGE_LOOP FACE_BOUND→VERTEX_LOOP (for loops with no edges)
Edge	ORIENTED_EDGE→EDGE_CURVE
B-curve	B SPLINE CURVE
Circle	CIRCLE
Ellipse	ELLIPSE
Parabola	BSPLINE CURVE
Hyperbola	B SPLINE CURVE
Line	LINE
Intersection curve	B SPLINE CURVE
SP-curve	B SPLINE CURVE
Trimmed curve	B SPLINE CURVE
Vertex	VERTEX POINT
Point	CARTESIAN POINT

5.2.2 Description of VPM Knowledge Classes

The high-level concepts and meta classes explained in the previous sections formulate the VPM knowledge classes in this research. The VPM knowledge classes are described and explained in Table 5-4. The VPM knowledge classes are developed to represent all the essential aspects that should be considered for modelling a product. In VPM, the captured design knowledge will be decomposed into these classes to describe the product model. The product model structure is also built through the atomic blocks named by the VPM knowledge classes. The VPM knowledge classes form the fundamental principle of the data exchange method in this research. All the captured non-geometric data is stored and

transferred through the use of a knowledge file which is developed based on the knowledge classes. Moreover, VPM knowledge classes also provide a steady data exchange format for knowledge store, reuse, and exchange within a knowledge-based product modelling environment. The implementation of the knowledge store and exchange is further explained in Section 5.3.3.

Table 5-4: Explanation of VPM knowledge classes.

Class	Description
Product	A Product means a physically realisable object that is produced by a process. It contains the product name and its identification number, and its description.
Feature	A Feature is a prominent aspect of a product that has a specific function. A product could have different design features. It can be described as architecture on the surface of the intended part with some engineering significance.
Function	A Function describes what the product is supposed to do.
Behaviour	Behaviour describes how the product implements its function.
Form	The form can be considered as aspects that distinguish the product by its substance, e.g., shape, contour, conformation, etc. In this product model, Form consists of four characteristics: geometry, material, mesh, and state.
Material	Material is the description of the physical substance that composites the product.
Design Intent	Design intent contains reasons or a logical basis for making or modifying a product.
Geometry	Geometry is the spatial description of the product.
Dimension	Dimension represents the Cartesian coordinates that determine a position of a product in space, e.g., length or width or height or radius.
Rules	Rules contain principles, standards and guidelines that govern the design of the product.
Fit	Fit represents the right size of the product that satisfies an assembly condition.
Constraint	A constraint is determined by design rules that restrict the product's spatial motion in an assembly.
Relationship	Relationship describes how two or more products are connected. In this VPM representation, it consists of two aspects which are Assembly and Reference.
Assembly	Assembly shows the hierarchical information of the product in an assembly. It can be further defined into "Parent" and "Children" to describe the assembly relationship.

Reference	Reference shows a mention or citation of the product.
Information	Information contains contexts collected in the product design that is not proven to be knowledge.

5.2.3 VPM Data Exchange Method and Format

The conducted literature in chapters 2 and 3 has identified that limitations still exist in the current STEP for exchanging parameters, design intent and other non-geometric data. To avoid the complex and tedious mapping process, data missing, and conversion errors, this research presents a new method for exchanging the data stored in the VPM product model (shown in Figure 5-8).

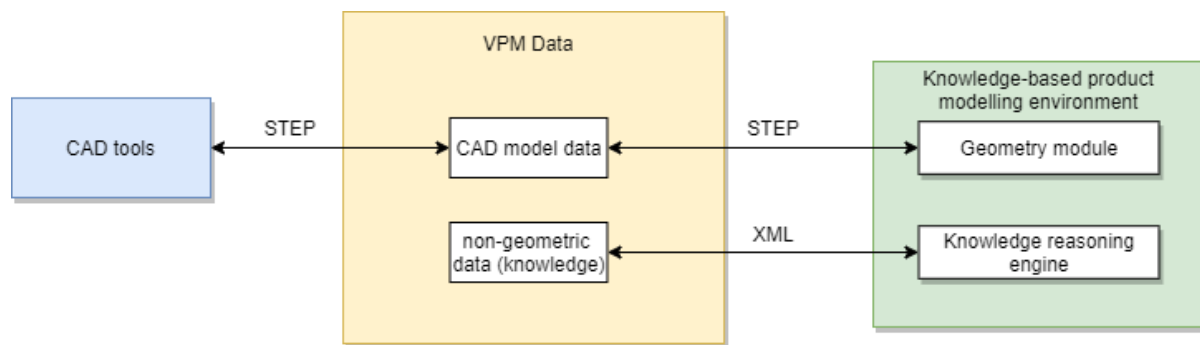


Figure 5-8: Description of VPM data exchange method

In the presented data exchange method, the STEP is only used for CAD model data exchange. This process could be done directly by using either a STEP translator in most existing CAD tools or a 3D data plugin. The captured knowledge (named non-geometric data) is stored and exchanged through the interoperable and platform-independent XML format. The implementation of this data exchange method is further explained in the below section.

5.3 Overall Virtual Product Modelling Framework Implementation Methods

As discussed in previous Section 5.2, the Virtual Product Modelling framework consists of five steps:

- Product model development
- Knowledge capture of non-geometric information
- Knowledge capture of geometric information
- Knowledge mapping
- Product visualisation & validation

This section will go through each step of this methodology and explain the techniques that are developed and utilised for the implementation of this Virtual Product Modelling framework in this research.

Section 5.1 shows how the knowledge classes are formulated and explains how a product model can be developed and represented with the knowledge classes in VPM to provide a generical product model structure. To provide data for this VPM product model, non-geometric information needs to be captured, classified and decomposed into the atomic blocks (as shown in Figure 5-2). The following sections 5.3.1 and 5.3.2 explain how the non-geometric information is captured and then describe which tool is chosen and utilised for the proof-of-concept implementation in this research.

5.3.1 Knowledge Source

To capture the non-geometric knowledge, it is essential to identify where the non-geometric knowledge is stored. In the current research, the knowledge applied by a design engineer in the modelling process is classified into four aspects: Standard, Specification, Experience, and Expertise. The standard defines the standardised discipline of how a product is modelled with generally accepted and uniform procedures, dimensions or materials. The specification is a detailed description of a customised product from the customer's point of view that describes

the design requirements in aspects such as function, dimensions, material, etc. Based on different specifications, product variants can be generated from a standardised product. Experience is the knowledge or skill obtained by the designers from the past modelling process, and Expertise is a higher level of the knowledge or skill acquired from previous experience, either theoretical or practical.

However, collecting, extracting, and verifying raw information and forming knowledge sources lie beyond the scope of the research. This research is based on the assumption that the knowledge of a product exists from the knowledge source.

5.3.2 Knowledge Capture

The term “knowledge capture” itself is a complex process that converts knowledge from tacit to explicit (Herschel, Nemati and Steiger, 2001). In the current research, knowledge capture is defined as a process that turns the knowledge from the existing knowledge source into an explicit representation with VPM knowledge classes.

After identifying the knowledge source, the next step is to find the suitable tool for capturing the non-geometric information. There are many PDM/PLM systems that provide the capabilities for collecting, retrieving and storing product data. However, those PDM/PLM systems are either heavy software packages that are enterprise-oriented and not free of charge or not capable of classifying the product data or exporting the data by following user-defined schema. In this research, a knowledge capture tool has been developed that allows designers to input the non-geometric information utilised in the product modelling process. The non-geometric information is then decomposed into the VPM classes, which are defined in the previous Section 5.2.2.

Since most web browsers have a built-in XML parser to access and manipulate XML, in this research, a web-based knowledge capture tool is developed using HyperText Markup Language (HTML), JavaScript and Hypertext Preprocessor (PHP). XML DOM (Document

Object Model) parser is used as a JavaScript to XML parsing method. This method presents an XML document as a clear tree structure and also enables programs to dynamically access and update the content and structure of the XML document. The web-based knowledge capture tool interface is shown in Figure 5-9.

The screenshot displays the 'Knowledge Capture Tool' interface. At the top, it says 'How many parts do you design?' with a dropdown menu set to '1 - Single Part' and a 'Back to Home Page' button. Below this, it prompts 'Start typing in the input field below:'. The main area is divided into two columns of input fields, each with a 'How many would you like to input?' dropdown menu.

Left Column:

- Product Information:** Product ID (with a note: 'if not known, a system id will be automatically created'), Product Name, Product Description, Product Type (with a note: '(part or assembly)'), Function, Material, Rule, Relationship, and Key Parameters.
- Design Intent:** Name, Description, Form, Behavior, Rule, Dimension, and Constraint.

Right Column:

- Design Intent:** Name, Description, Form, Behavior, Rule, Dimension, and Constraint.
- Form:** Name, Description, Property.
- Behavior:** Name, Description, Property.
- Rule:** Name, Description, Property.
- Dimension:** Length, Width, Height, Radius, Diameter, and Constraints.
- Constraint:** Name, Description, Property, and Constraint type (with a note: '(select if known)').

Each input field is accompanied by a small dropdown menu for the number of items to capture. A 'Submit' button is located at the bottom left of the form area.

Figure 5-9: The developed knowledge capture tool interface (maximised view of this tool interface is provided in Appendix)

5.3.3 Knowledge Store and Exchange

All the captured non-geometric information from the knowledge capture tool is stored in an XML document. To ensure the data structure of the XML document, an XML schema has

been developed based on the framework of VPM. This XML document is named Knowledge File (KF) in this research and will be used for data exchange. The knowledge file contains all the non-geometric information from the VPM, including the dimensions, key parameters, and design rules of the product. An example of a simplified XML schema for the Knowledge File data is shown in Figure 5-10.

```
<knowledge>
  <product type="part">
    <id></id>
    <name></name>
    <description></description>

    <design_intent> ... </design_intent>

    <function_> ... </function_>

    <form> ... </form>

    <material> ... </material>

    <behaviour> ... </behaviour>

    <fit> ... </fit>

    <relationship> ... </relationship>

    <rules> ... </rules>

    <dimension> ... </dimension>

  </product>
</knowledge>
```

Figure 5-10: Example of a simplified XML schema for the Knowledge File

5.3.4 Knowledge Mapping

To establish interaction between the non-geometric information and geometry and provide knowledge reasoning in the product modelling process, rules and key parameters stored in the knowledge file are used to map with the product's dimension. Meanwhile, other aspects of the VPM can also be constrained depending on the rule's scope. Figure 5-11 shows a knowledge mapping logic example of how a dimensional parameter can be mapped with the knowledge. In this example, an M12 hex bolt has two parameters: b (thread length) and L

(bolt length). The existing standard has constrained the two parameters by the following rules:

- 1) if L is less than 125 mm, then thread length b should be 30 mm.
- 2) if L is between 125 mm and 200 mm, the thread length b should be 36 mm.
- 3) if L is larger than 200 mm, the thread length b should be 49mm.

Therefore, this rule can be used to build the interaction between the dimensional parameters b and L with the product modelling process (changing parameter b). Further, this rule also provides the information for knowledge reasoning when b is adjusted to different values. This process is performed within an object-oriented programming environment that automatically selects the VPM knowledge classes, parses the stored data and shows all the reading results.

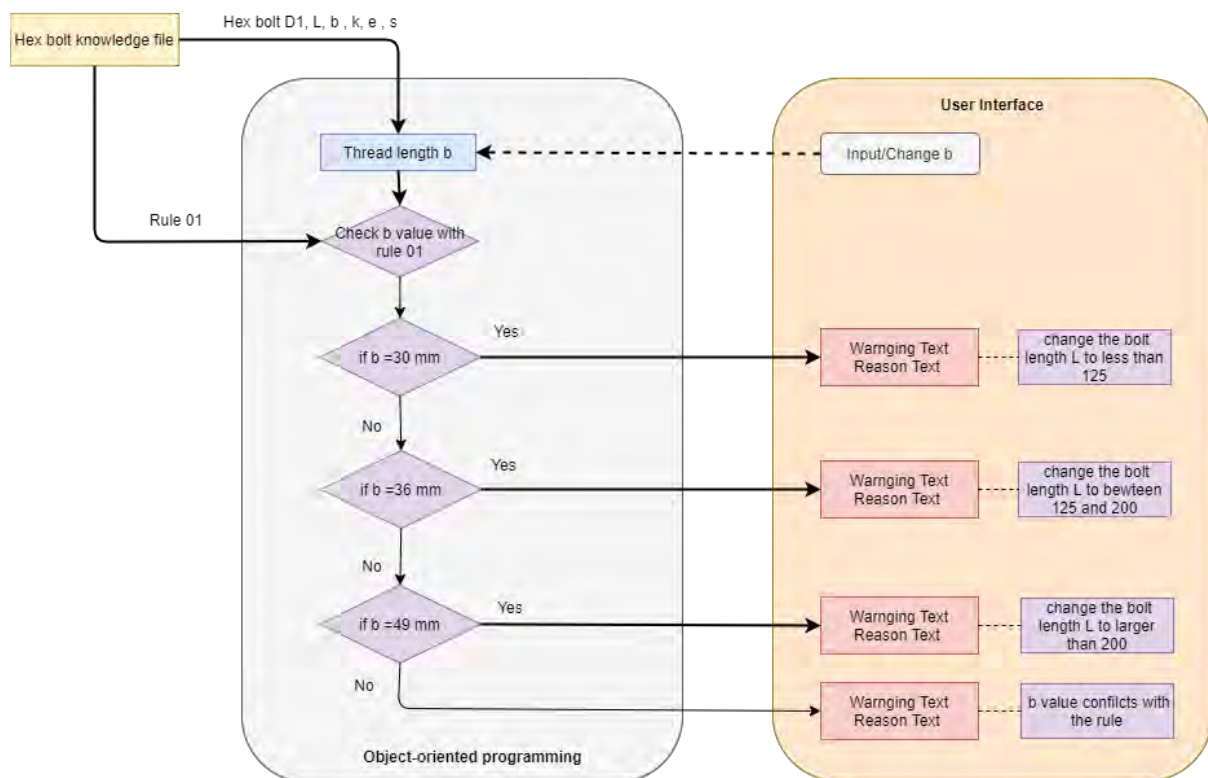


Figure 5-11: Example of knowledge mapping logic for implementation

5.3.5 Visualisation

The visualisation provides the ability to display the product model's original geometry and the possible changes that can be made, and the associated knowledge that constrains the

changes. According to the previous discussion, Unity3D is selected as the development tool to develop a knowledge-based product modelling environment as a proof-of-concept tool, and C# is used as an object-oriented programming language.

The first type of visualisation is shown in the tool by importing the geometry file of the product into the interface through the Unity plugin. It provides a 3D view of the geometry of the product model. An example is shown in Figure 5-12 (a). The second type of visualisation is the text representation of all the non-geometric information stored in the knowledge file (shown in Figure 5-12 (b)). This process is performed with object-oriented programming that automatically analyses and displays all the acquired non-geometric information that is stored in the knowledge file. The developed visualisation algorithm is performed by searching through the XML tags. The third type of visualisation is the possible change of geometry. However, due to the nature of STEP, there are no existing technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. To develop such a method that can manipulate and visualise STEP files is beyond the scope of this research. Instead of visualising the graphical changes in the geometry, this research uses text visualisation to indicate the changes that are made to the geometry (as shown in Figure 5-12 (c)).

The final stage of visualisation is to show the associated knowledge that is applied to make or constrain the change of geometry. An example is shown in Figure 5-12 (d). This includes the following:

- Dimensions that are affected by the changes,
- Key parameters that are affected by the changes,
- Design rules that constrain the changes,
- Other aspects that are affected by the changes.

The visualisation of the hex bolt model example in Figure 5-12 includes: (a) 3D view of the bolt geometry and (b) visualisation of all the non-geometric information stored in the bolt knowledge file, (c) Functions of making possible changes of bolt geometry, (d) visualisation of the associated knowledge that is applied to constrain the change of geometry.

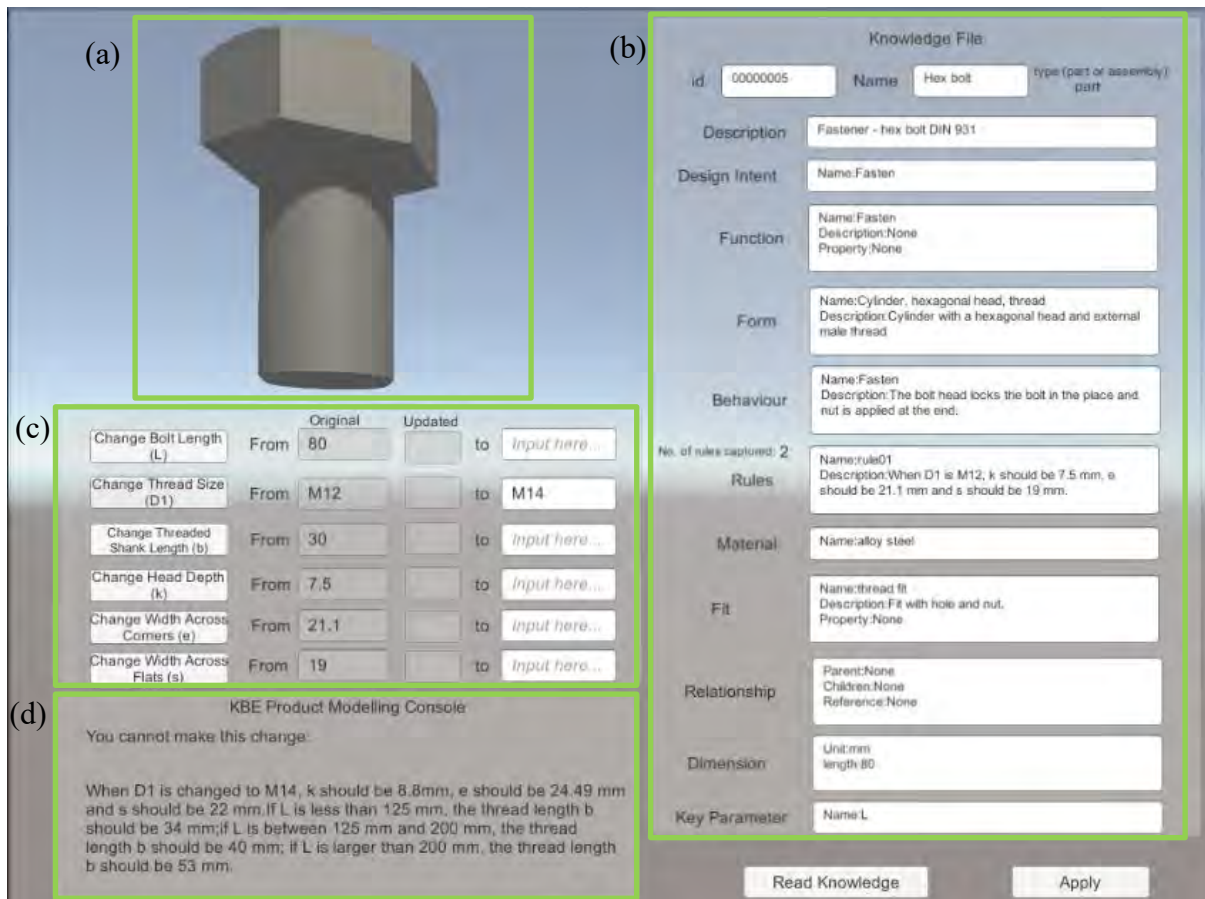


Figure 5-12: Example of visualisation in the developed knowledge-based product modelling environment

5.4 Chapter Summary

This chapter presented and discussed each stage of the virtual product modelling framework development. It explained the development of the product model in VPM and provided an explanation of the VPM knowledge classes. The formulated VPM knowledge classes were used to build the atomic product model structure in VPM and to form the knowledge store

and exchange method used in this research. A knowledge schema was developed to provide a steady format for knowledge exchange. In this research, the captured knowledge will be stored and transferred using a knowledge file (under the knowledge schema) which was created by using the developed knowledge capture tool. Further, the implementation methods of this framework were provided based on the tool availability and research needs. The developed product model structure, selected tools and formats, and the developed data exchange methods were used to develop the knowledge-based product modelling environment as a proof-of-concept tool. The five stages of VPM will be implemented with use cases to validate the proposed framework. The following chapter presents the use case evaluation of VPM by using the developed knowledge-based product modelling environment.

6 Evaluation

6.1 Introduction

In this research, this virtual product modelling framework is implemented through the development of a knowledge-based product modelling system. As discussed in previous chapters, limited studies have been done to implement and evaluate KBE frameworks. This research addresses this identified research gap by not only providing a virtual product modelling framework but also implementing the whole system within different use cases and analysing the effectiveness and efficiency.

This chapter explains the evaluation of the virtual product modelling framework through three test use cases. It provides detailed instantiation steps of this virtual product modelling framework. In the development process of an interactive engineering system, use cases are typically relevant to two key aspects, which are system development and user interface design (Hornbæk *et al.*, 2007). In the current research, system development is considered as the development of the backend of the knowledge-based product modelling environment. It includes the development of the product model itself and the development of the rules and relationships between different atomic blocks in the VPM. User interface design is considered as the development of the frontend of the modelling system. A product modelling user interface is developed as the frontend to visualise the modelling process and for knowledge representation.

As discussed in Section 4.3.4 of Chapter 4, use case evaluation is widely adopted to evaluate various design products such as virtual models, prototypes, scenarios, systems, and interfaces. In this chapter, three test use cases are selected to demonstrate and evaluate the effectiveness and efficiency of the proposed methodology. Existing knowledge gathered in the use cases is applied for the evaluation of the framework. Critical analysis and comparison between

existing/legacy product modelling systems and the VPM framework are also provided. Discussion and findings from the use case evaluation are presented at the end of this chapter.

6.2 Evaluation Objectives

The objectives of the evaluation process are identified as follows:

- Perform different scenarios in the use cases to compare the actual results from applying virtual product modelling framework for capturing and reusing design knowledge against the identified evaluation criteria.
- Analyse the virtual product modelling framework results, compare, and contrast the product modelling results from using the virtual product modelling framework with the use of the current existing/legacy product modelling system for the same circumstances.

As mentioned above, since there have been limited studies in implementing and evaluating the KBE system, the evaluation criteria have been developed in Section 4.4.4 of Chapter 4 from the key challenges that have been identified in the existing CAD systems and KBE methodologies. In order to prove the effectiveness and efficiency of the proposed methodology and to analyse the difference in product modelling between current product modelling methodologies/legacy product modelling systems and the virtual product modelling methodology evaluation results, measurement parameters (shown in Table 6-1) have been set up based on the evaluation criteria, and the findings from the literature review synthesis (Section 3.8.3) of existing methodologies and relevant research work. The parameters that will be used to measure the workability and effectiveness of the framework are knowledge capture, product geometry and knowledge visualisation, product relationship representation, knowledge reasoning and reuse, the correctness of the changes, data exchange of geometry and data exchange of knowledge. The measurement parameters are further explained in the later sections for each use case.

Table 6-1: Measurement parameters mapped with evaluation criteria

Measurement parameter	Parameter description	Evaluation criteria	Criteria description
Generative representation	If the VPM can develop a model as a generative representation of the product	C1	The capability of generative representation of engineering products in VPM
Knowledge capture	If the knowledge capture tool can capture the existing product information as existing knowledge and generate a knowledge file	C2	The capability of the VPM to capture the product geometry and its associated knowledge from the existing product information
Product geometry and knowledge visualisation	If the interface can visualise the part geometry and its associated knowledge	C3	The capability of the VPM to visualise the product geometry and its associated knowledge
Product relationship representation	If the interface can show the relationship between part-part, part-assembly	C4	The capability of the VPM to present every part of the product and the relationships among them
Knowledge reasoning and reuse	If the product geometry is constrained by rules when the users change the dimension of the part in the product modelling process	C5	The capability of the VPM to propagate changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge
Correctness of the changes	If the interface can drive and constrain the	C6	The correctness of the changes applied to the

	change of part dimension through rules and propagate the changes correctly.		product geometry by reuse of the existing knowledge
Data exchange of geometry	If the geometry file can be exchanged through STEP file with CAD platform.	C7	The capability and correctness of the product geometry data exchange between different platforms
Data exchange of knowledge	If the knowledge file can be exchanged. If the interface can propagate the changes of knowledge in the knowledge file after modifying the knowledge file and re-importing.	C8	The capability and correctness of the knowledge exchange through knowledge file

The evaluation through use case is based on the data collected from the literature during the literature review. The first use case is adapted from the primitive design feature examples for the basic engineering feature modelling (Leu, 2016). This use case is selected because primitive design features are the fundamental geometric features applied in the actual product modelling process in general CAD environments. The second use case is a bolt example from literature. Since bolts and nuts are the most basic components for mechanical engineering, modelling these components can describe how engineering rules are applied in basic engineering part modelling on a conceptual level. The third use case is a wheel assembly example from literature. A wheel assembly with a wheel part and a tyre part is selected to describe how two engineering parts are connected and constrained in one engineering assembly.

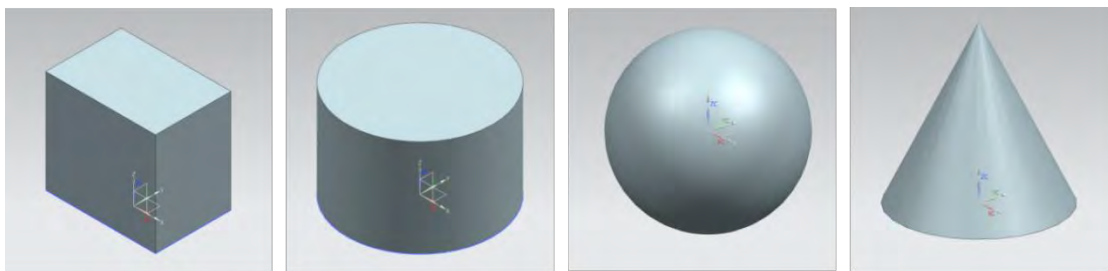
As explained in previous Section 3.8.3, the virtual product modelling framework is developed for capturing and reusing existing knowledge to support design engineering automation. Therefore, when applying the virtual product modelling framework to one product, it is assumed that the product model has already been created in the CAD software. So, before the evaluation starts, all the use cases are pre-modelled in one of the current product modelling legacy systems – Siemens NX 10 and the existing information are collected manually while modelling these use cases. All these product models and existing information are then reused as existing knowledge in the evaluation process. To evaluate the tool’s effectiveness, it is also assumed that the existing information contains sufficient knowledge concerning the virtual product modelling classes for the identified testing scenarios in each use case. Furthermore, to specify a product or constraint different aspects of a product in the product modelling process, design rules can be in various forms, such as text, equation, formula, etc. Nevertheless, when performing these design rules through object-oriented programming, they all constrain the product with programming logic. In other words, in the evaluation process of this research, rules are converted into algorithms with programming logic, and all of the VPM classes representing a product are treated as programming parameters, and these parameters will be constrained by programming logic. Different rules are defined manually for each use case to test the tool’s effectiveness with different scenarios. These rules are distinguished as:

- Rules that constrain a single parameter in one part. (Use Case 1)
- Rules that constrain parameters in one part. (Use Case 2)
- Rules that constrain parameters between parts in one assembly (Use Case 3)

6.3 Use Case 1: Simple Part with Primitive Design Features

6.3.1 Use Case Overview

In the first use case, four simple parts are selected based on the primitive design feature example from literature (Leu, 2016) to evaluate different aspects of the proposed virtual product modelling framework. Primitive features are basic geometric features from which many other design features can be created. The basic primitive design features are block, cylinder, sphere and cone. In this use case, the virtual product modelling approach is applied to four simple parts that are formed from these primitive design features, which are a block part, a cylinder part, a cone part, and a sphere part accordingly (shown in Figure 6-1). Existing information of each simple part is discussed separately in sections 6.3.2, 6.3.3, 6.3.4 and 6.3.5.



(a) a block part (b) a cylinder part (c) a sphere part (d) a cone part

Figure 6-1: Four simple parts with primitive design features (modelled in Siemens NX 10)

6.3.2 Simple Part – a Block Part

In this use case, existing information is collected from the part library of one of the current product modelling systems – Siemens NX 10. However, the Siemens NX part library does not provide all the information that fits the classified VPM classes. Hence, for some VPM classes, such as material, behaviour, fit, and relationship, the entities are given as “None” or “Not defined”.

Rules defined for the testing scenarios are regarded as the existing design rules. A list of information of the block part is shown in the following Table 6-2. The listed product information is assumed to be the non-geometric information that will be captured for VPM.

Table 6-2: Existing information of use case 1- primitive design feature: block part

VPM knowledge class	Existing product information
Product	Block
Feature	Primitive design feature - block
Description	The Block is a cube.
Function	None
Behaviour	None
Form	Geometry: block
Material	Not defined
Design intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Length =100mm, width =100mm, height=100mm
Rules	Block Length L = Block Width W = Block Height H
Fit	None
Constraint	None
Relationship	None
Reference	None

6.3.2.1 Measurement Parameters and Testing Scenarios

Before applying the developed framework to the first use case, a list of measurement parameters is set up based on the evaluation criteria (as shown in Table 6-3). Moreover, to verify and validate the tool's effectiveness, testing scenarios are also defined. The scenario identified for this block part is "single dimension changed (block length) with single rule applied".

Table 6-3: Measurement parameters and expected results of use case 1

Measurement parameter	Evaluation criteria	Explanation	Expected results
Generative representation	C1	If the VPM can develop a model as a generative representation for each simple part.	VPM product model structure
Knowledge capture	C2	If the knowledge capture tool can capture the existing part information as existing knowledge and generate a knowledge file for the part.	Knowledge file generated in XML format for each simple part
Product geometry and knowledge visualisation	C3	If the interface can visualise each simple part geometry (from the step file) and its associated knowledge (from the knowledge file)	Geometry visualisation of the part in the interface. Knowledge visualisation of the knowledge file in the interface.
Product relationship representation	C4	There is no relationship between the part and other parts because it is a single part and does not belong to any assembly	The product is shown as a “part”
Knowledge reasoning and reuse	C5	If each simple part dimension is driven and constrained by rules with knowledge reasoning.	Constrain the change of dimension by rules with knowledge reasoning
Correctness of the changes	C6	If the changes applied to the part geometry through reuse of the existing knowledge are correct	Propagate the change of dimension correctly
Data exchange of geometry	C7	If the geometric data of each simple part is exchanged through the STEP file	Geometry visualisation in the interface
Data exchange of knowledge	C8	If the captured knowledge can be exchanged through a knowledge file.	Modify the knowledge file, re-import it and present the changes of knowledge in the knowledge file

6.3.2.2 Virtual Product Modelling Framework Application

After the existing information is collected, the next step is to utilise the current product model data as the input to evaluate the virtual product modelling framework. The following sections will focus on each step of the proposed virtual product modelling framework applied to this use case - block.

a) Product model development

As explained in Chapter 5, a product model for representing a general product has been developed in this research. Thus, in this use case, the block part can be represented using the VPM product model structure to provide a clear picture of the atomic decomposition of product knowledge (shown in Figure 6-2).

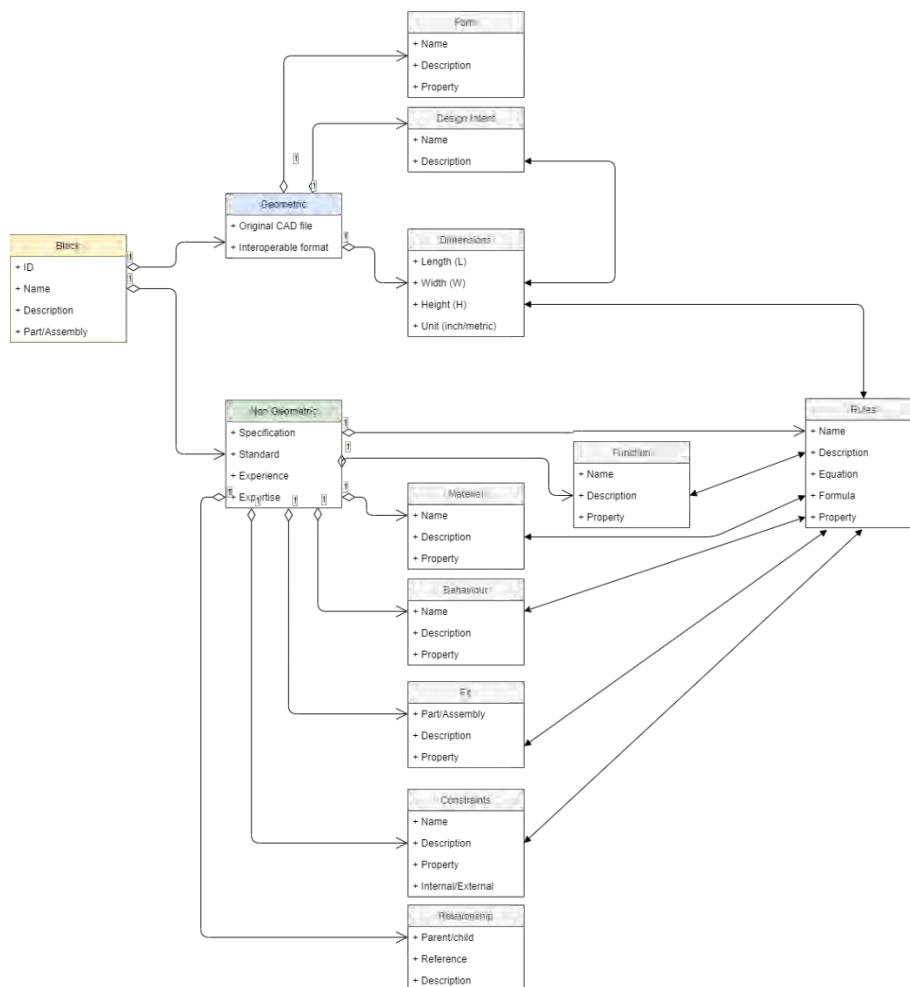
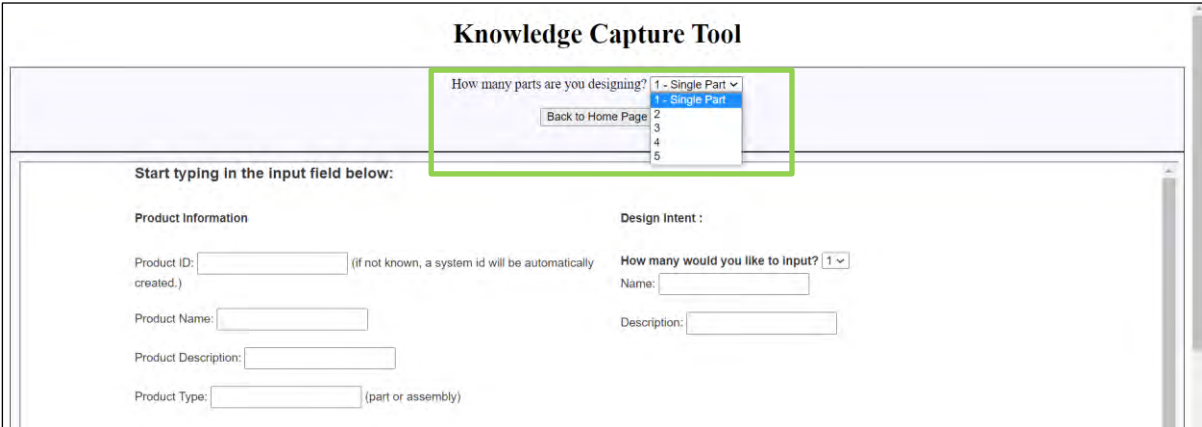


Figure 6-2: VPM product model structure of the block part in UML diagram

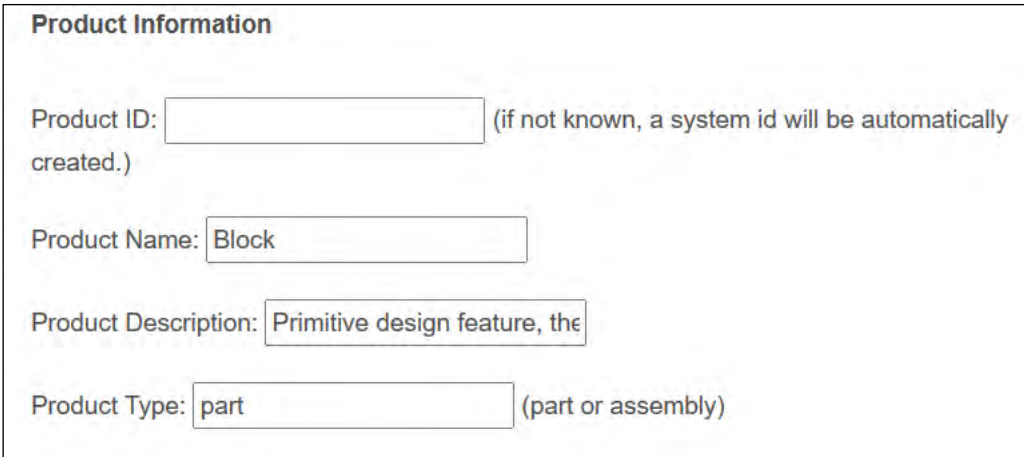
b) Knowledge capture of non-geometric information

In this step, all the existing information is regarded as design knowledge and captured using the knowledge capture tool introduced in Section 5.3.2. The knowledge capture tool guides the users to input their design knowledge by asking them to provide information on different aspects of the product. This is a standard process that is required in the knowledge capture of non-geometric information step for each use case. The following figures from Figure 6-3 to Figure 6-15 explain each step of the knowledge capturing of non-geometric information of the block part as examples of this process. The maximised view of the knowledge capture tool is provided in the Appendix.



The screenshot shows the 'Knowledge Capture Tool' interface. At the top, there is a header with the title 'Knowledge Capture Tool'. Below the header, there is a section titled 'How many parts are you designing?' with a dropdown menu. The dropdown menu is open, showing options: '1 - Single Part', '2', '3', '4', and '5'. The first option, '1 - Single Part', is selected. A 'Back to Home Page' button is located below the dropdown. Below this section, there is a large input field with the prompt 'Start typing in the input field below:'. The main form area is divided into two columns: 'Product Information' and 'Design Intent:'. The 'Product Information' column contains four input fields: 'Product ID: [] (if not known, a system id will be automatically created.)', 'Product Name: []', 'Product Description: []', and 'Product Type: [] (part or assembly)'. The 'Design Intent:' column contains two input fields: 'How many would you like to input? [1]' and 'Name: []', followed by a 'Description: []' field.

Figure 6-3: Example - select the number of the parts being designed



The screenshot shows the 'Product Information' section of the Knowledge Capture Tool. It contains four input fields: 'Product ID: [] (if not known, a system id will be automatically created.)', 'Product Name: [Block]', 'Product Description: [Primitive design feature, the]', and 'Product Type: [part] (part or assembly)'. The 'Product Name' field is filled with 'Block', the 'Product Description' field is filled with 'Primitive design feature, the', and the 'Product Type' field is filled with 'part'.

Figure 6-4: Example - input the Product Information including name, description, and type

Design Intent :

How many would you like to input?

Name:

Description:

- 1
- 2
- 3
- 4
- 5

(a) Select the number of input

Design Intent :

How many would you like to input?

Name:

Description:

(b) Input the name and description

Figure 6-5: Example - input the Design Intent and its description

Function :

How many would you like to input?

- 1
- 2
- 3
- 4
- 5

Name:

Description:

Property:

(a) Select the number of input

Function :

How many would you like to input?

Name:

Description:

Property:

(b) Input the information about the Function

Figure 6-6: Example - input the Function

Form :

How many would you like to input?

- 1
- 2
- 3
- 4
- 5

Name:

Description:

(a) Select the number of input

Form :

How many would you like to input?

Name:

Description:

(b) Input the information about the Form

Figure 6-7: Example - input the Form

Material:

How many would you like to input?

1 ▾

1
2
3
4
5

Name:

Description:

Property:

(a) Select the number of input

Material:

How many would you like to input?

1 ▾

Name:

Description:

Property:

(b) Input the information about the Material

Figure 6-8: Example - input the Material

Behaviour :

How many would you like to input?

1 ▾

1
2
3
4
5

Name:

Description:

Property:

(a) Select the number of input

Behaviour :

How many would you like to input?

1 ▾

Name:

Description:

Property:

(b) Input the information about the Behaviour

Figure 6-9: Example - input the Behaviour

Rules :

How many would you like to input?

1 ▾

1

2

3

4

5

Description:

Equation:

Formula:

Property:

(a) Select the number of Rules

Rules :

How many would you like to input?

1 ▾

Name:

Description:

Equation:

Formula:

Property:

(b) Input the Rules

Figure 6-10: Example - input the Rules

Fit:

How many would you like to input?

1 ▾

1

2

3

4

5

Description:

Property:

(a) Select the number of input

Fit:

How many would you like to input?

1 ▾

Name:

Description:

Property:

(b) Input the information about Fit

Figure 6-11: Example - input the Fit

Relationship:

How many would you like to input?

1 ▾

1
2
3
4
5

Parent:

Children:

Reference:

Description:

Relationship:

How many would you like to input?

1 ▾

Parent:

Children:

Reference:

Description:

(a) Select the number of input

(b) Input the information about Relationship

Figure 6-12: Example - input the Relationship

Dimensions:

How many would you like to add?

1 ▾

Length: Unit:

Width: Unit:

Height: Unit:

Radius: Unit:

Diameter: Unit:

Figure 6-13: Example - input the Dimension

Key Parameters

How many would you like to input?

3 ▾

1
2 :
3
4 :
5

Unit:

Name:

Value:

Unit:

Name:

Value:

Unit:

Key Parameters

How many would you like to input?

3 ▾

Name:

Value:

Unit:

Name:

Value:

Unit:

Name:

Value:

Unit:

(a) Select the number of Key Parameters

(b) Input the Key Parameters

Figure 6-14: Example - input the Key Parameters

Constraints

How many would you like to input?

1 ▾

Name:

Description:

Property:

Constraint type: (internal or external)

Figure 6-15: Example - input the Constraints

After the existing information has been input into the knowledge capture tool, a knowledge file in the format of XML will be generated. This knowledge file stores all the captured non-geometric information (shown in Figure 6-16).

```

<knowledge>
  <product type="part">
    <id>00000001</id>
    <name>Block</name>
    <description>Primitive Design Feature, the block is a cube.</description>

    <design_intent>
      <name>Primitive design feature</name>
      <description>To create other design features</description>
    </design_intent>

    <function_>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </function_>

    <form>
      <name>Block</name>
      <description>Geometry: Block.</description>
    </form>

    <material>
      <name>not defined</name>
      <description>None</description>
      <property>None</property>
    </material>

    <behaviour>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </behaviour>

    <fit>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </fit>

    <relationship>
      <parent>None</parent>
      <children>None</children>
      <reference>None</reference>
      <description>None</description>
    </relationship>

    <rules>
      <name>rule01</name>
      <description>Block Length L = Block Width W = Block Height H</description>
      <formula>L=W=H</formula>
      <equation>L=W=H</equation>
    </rules>
  </product>
</knowledge>

```

Figure 6-16: Example - part of knowledge file of the block part

By filling the captured knowledge into the atomic blocks of the VPM product model structure, the block part can be further represented with supplementary knowledge (shown in Figure 6-17).

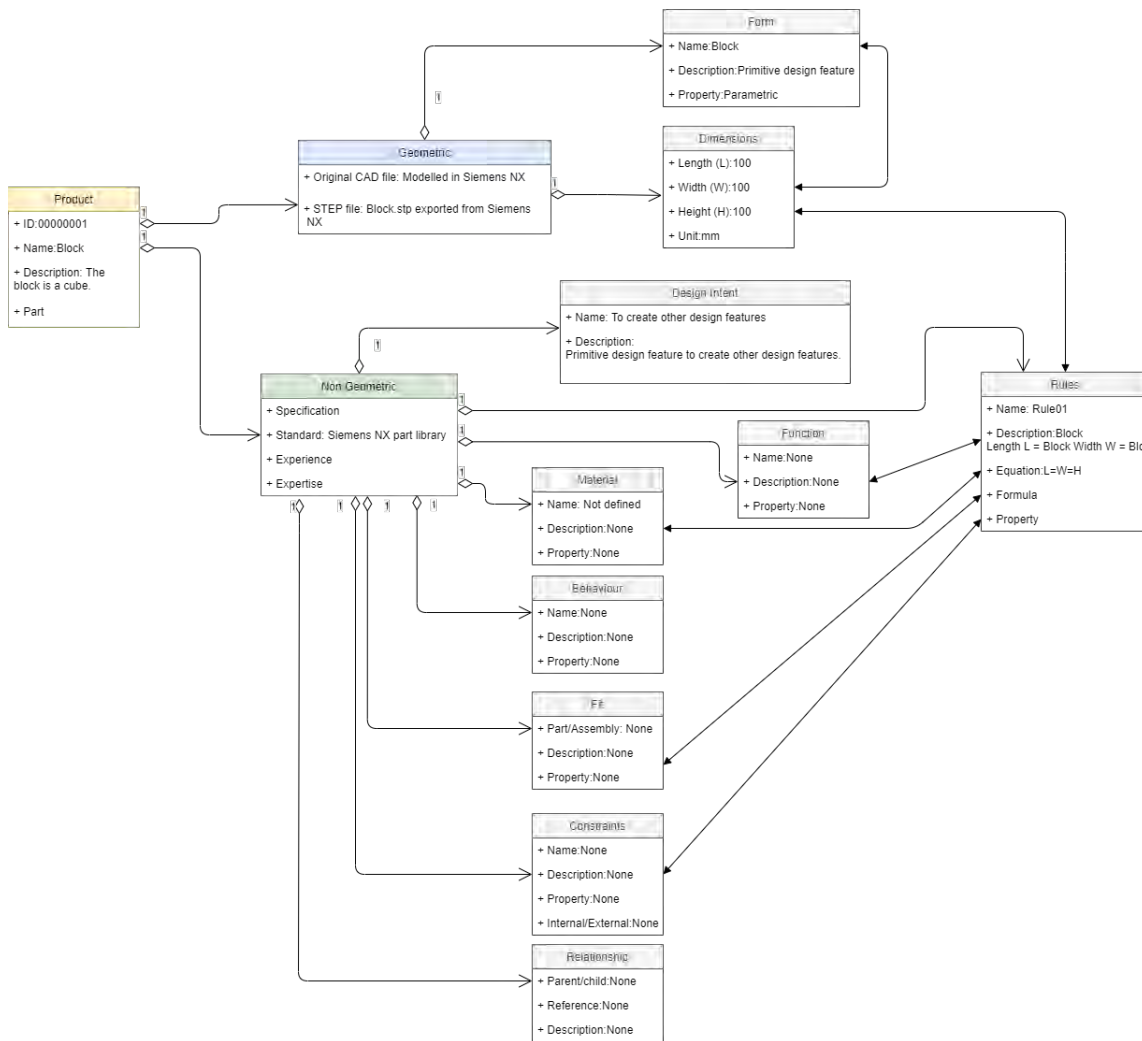


Figure 6-17: VPM product model structure of the block part with captured knowledge

c) Knowledge capture of geometry information

To visualise the product represented by the VPM, geometry information of the part needs to be extracted and stored in a STEP file. This is a typical process that is required in the knowledge capture of geometry information step for each use case. This process is done through the following steps:

- Pre-model the part in a CAD software Siemens NX 10 (example of a block part is shown in Figure 6-18)
- Exported the part into a STEP file (Figure 6-19) using the export function in Siemens NX 10.

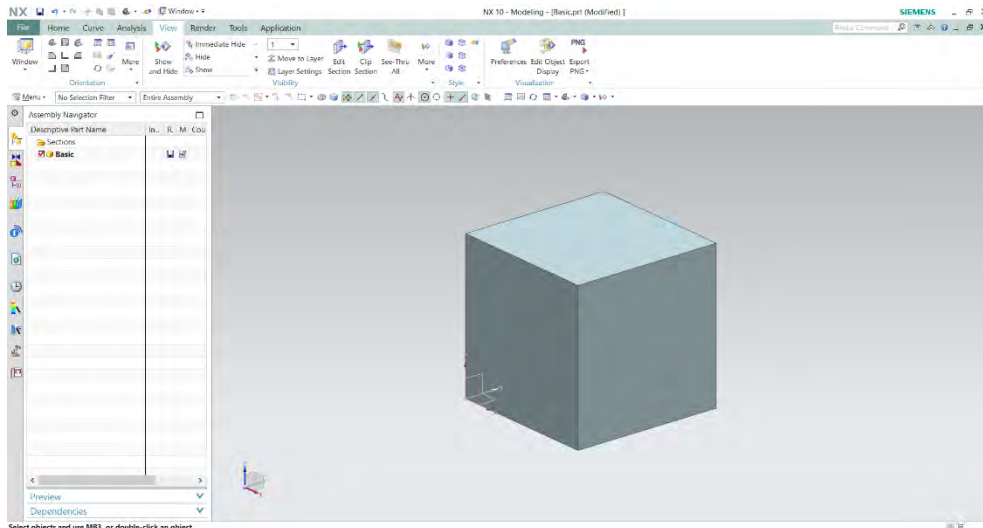


Figure 6-18: A block part model created in Siemens NX 10

```

Block - Notepad
File Edit Format View Help
DATA;
#10=SHAPE_REPRESENTATION_RELATIONSHIP('None',
'relationship between Basic-None and Basic-None',#92,#11);
#11=ADVANCED_BREP_SHAPE_REPRESENTATION('Basic-None',(#93),#247);
#12=CC_DESIGN_APPROVAL(#24, (#82));
#13=CC_DESIGN_APPROVAL(#25, (#84));
#14=CC_DESIGN_APPROVAL(#26, (#31));
#15=APPROVAL_PERSON_ORGANIZATION(#69, #24, #18);
#16=APPROVAL_PERSON_ORGANIZATION(#70, #25, #19);
#17=APPROVAL_PERSON_ORGANIZATION(#71, #26, #20);
#18=APPROVAL_ROLE('approver');
#19=APPROVAL_ROLE('approver');
#20=APPROVAL_ROLE('approver');
#21=APPROVAL_DATE_TIME(#39, #24);
#22=APPROVAL_DATE_TIME(#40, #25);
#23=APPROVAL_DATE_TIME(#41, #26);
#24=APPROVAL(#27, ' ');
#25=APPROVAL(#28, ' ');
#26=APPROVAL(#29, ' ');
#27=APPROVAL_STATUS('not_yet_approved');
#28=APPROVAL_STATUS('not_yet_approved');
#29=APPROVAL_STATUS('not_yet_approved');
#30=CC_DESIGN_SECURITY_CLASSIFICATION(#31, (#84));
#31=SECURITY_CLASSIFICATION(' ', '#32);
#32=SECURITY_CLASSIFICATION_LEVEL('confidential');
#33=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#37, #35, (#82));
#34=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#38, #36, (#31));
#35=DATE_TIME_ROLE('creation_date');
#36=DATE_TIME_ROLE('classification_date');
#37=DATE_AND_TIME(#52, #42);
#38=DATE_AND_TIME(#53, #43);
#39=DATE_AND_TIME(#54, #44);
#40=DATE_AND_TIME(#55, #45);
#41=DATE_AND_TIME(#56, #46);
#42=LOCAL_TIME(0, 0, 0, #47);
#43=LOCAL_TIME(0, 0, 0, #48);
#44=LOCAL_TIME(0, 0, 0, #49);
Ln 39, Col 25    100%    Windows (CRLF)    UTF-8

```

Figure 6-19: Part of the step file of block that exported from Siemens NX 10

d) Knowledge mapping

The rules stored in the knowledge file are used to map with the block dimension to establish the link between the geometry information and the knowledge. In this block part use case, the rule is defined as “Block Length (L) = Block Width (W) = Block Height (H)”. This mapping is done through object-oriented programming in the implementation tool Unity. Thus, when

the user is trying to change the length of the block in the interface, the algorithm will run and tell the user that the changes are affected by the defined rule. Figure 6-20 shows the mapping process between the knowledge file and the user interface for visualisation. The detailed codification can be found in Appendix.

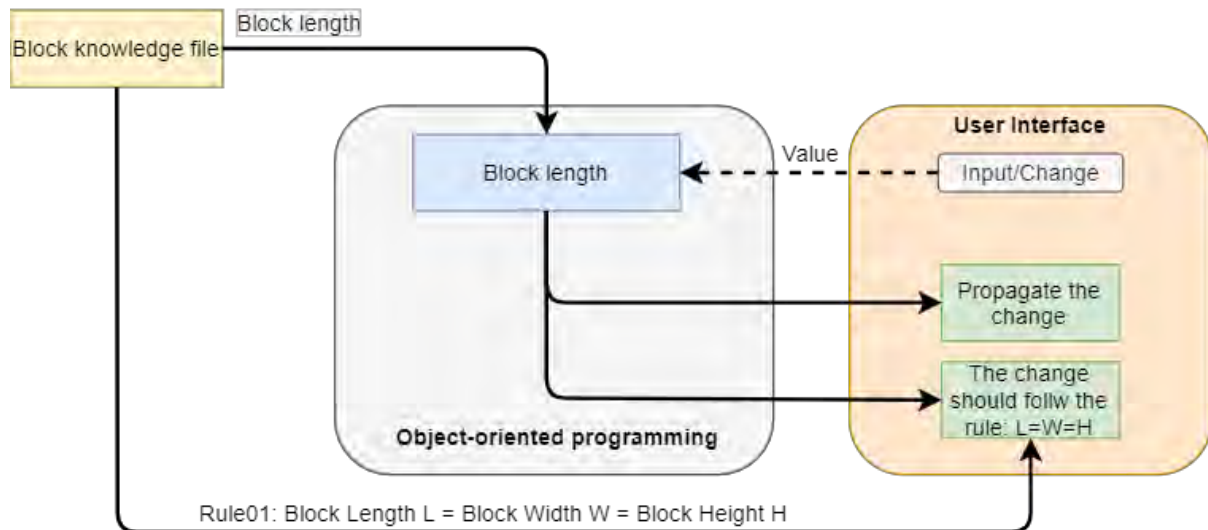


Figure 6-20: Illustration of the knowledge mapping for the block part in use case 1

e) Visualisation & validation

The visualisation of results is performed both in text and 3D visualisation. After importing the STEP file of the block part into the user interface, a 3D model of the block can be visualised. By clicking the “Read Knowledge” button, the developed tool will automatically parse the knowledge file and display all the captured information in the interface (see Figure 6-21).

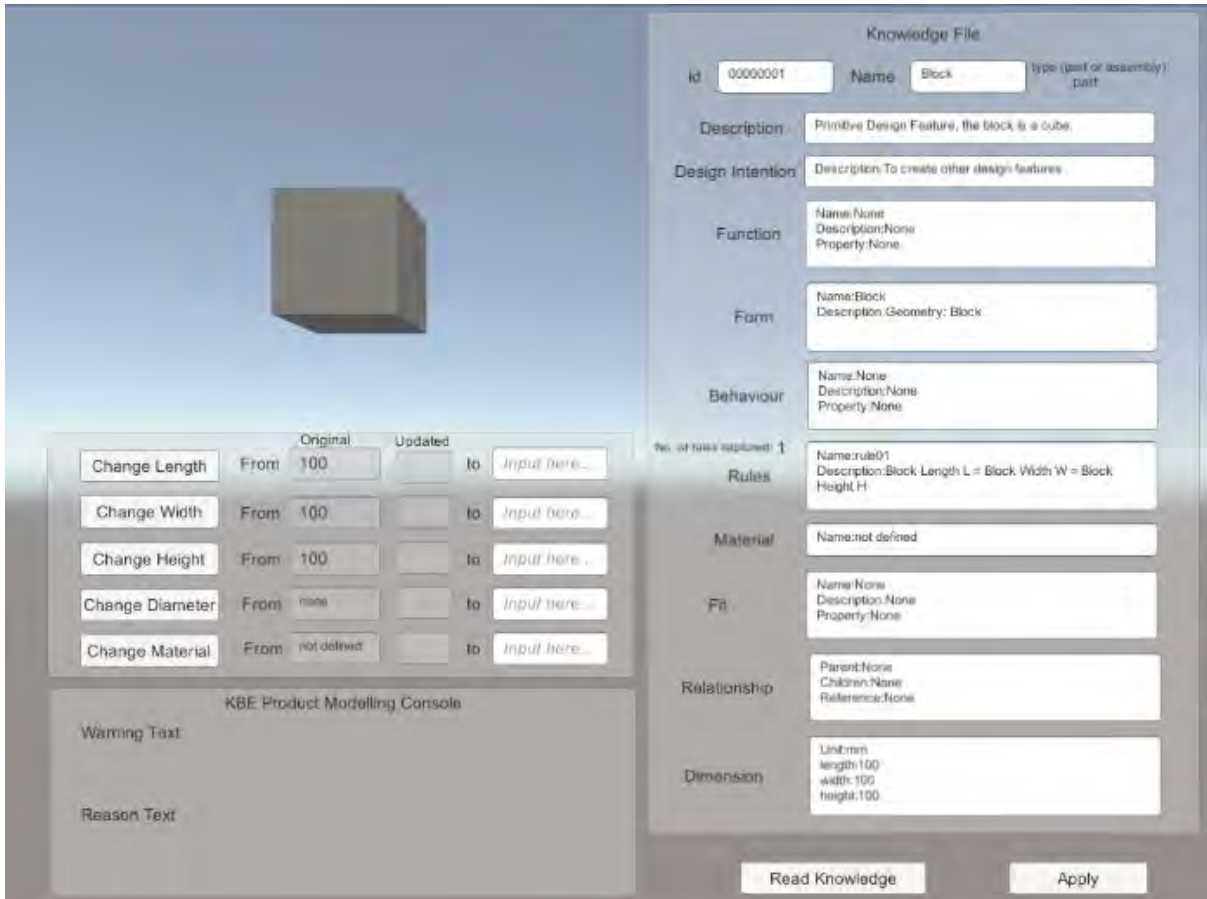
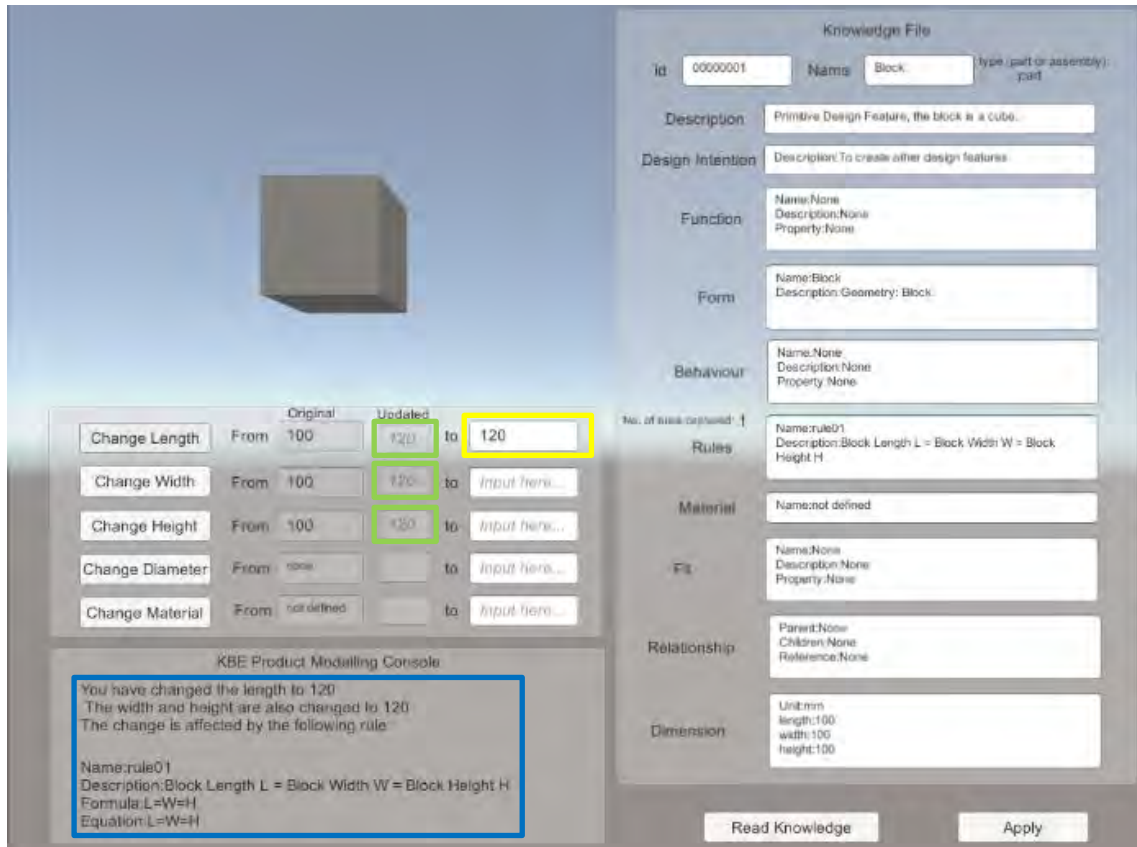


Figure 6-21: Block part model visualised in the developed knowledge-based product modelling environment

To validate the tool’s effectiveness, one testing scenario is defined previously in Section 6.3.2.1, which is “single dimension changed (block length) with single rule applied”. After the user selects to change the block length by clicking the button “Change Length” in the tool interface, the resulting changes will be shown in the “KBE Product Modelling Console” panel. The results for this testing scenario with a single dimension changed are shown in Figure 6-22. In the meantime, the “KBE Product Modelling Console” also indicates the rule affecting the changes. For instance, when the user input 120 in the interface to change the length from the original 100 to 120 (unit: mm), the “KBE Product Modelling Console” will analyse if this change can be made by checking the rules. In this testing scenario, the rule that has been applied to the block part is “Length(L) = Width(W) = Height (H)”. Therefore, the

width and height of the block are also changed to 120 automatically. And the affecting rule is shown correctly in the “KBE Product Modelling Console”, which fulfils the knowledge reasoning for this product modelling process. These results will be further discussed and analysed at the end of this use case in Section 6.3.6.



Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-22: Results of validation – use case 1: Block part

6.3.3 Simple Part – a Cylinder Part

Similar to the previous block part, the existing information of a simple cylinder part is collected from the part library of Siemens NX 10. Information such as material, behaviour, fit, and relationship is given as “None” or “Not defined” for this evaluation. A list of the cylinder part information is shown in Table 6-4. The listed product information is assumed to be the non-geometric information that is going to be captured for VPM.

Table 6-4: Existing information of use case 1- primitive design feature: cylinder part

VPM knowledge class	Existing product information
Product	Cylinder
Feature	Primitive design feature - cylinder
Description	Cylindrical part
Function	None
Behaviour	None
Form	Geometry: Cylinder
Material	Not defined
Design intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Diameter =50 mm, Height = 100mm.
Rules	Rule 01: The height of cylinder should not be larger than 200. Rule 02: The diameter of cylinder should not be larger than 80.
Fit	None
Constraint	None
Relationship	None
Reference	None

6.3.3.1 Measurement Parameters and Testing Scenarios

Same measurement parameters for evaluation with use case 1 are used for this cylinder part (as shown in Table 6-3). Testing scenarios are also defined to verify and validate the tool's effectiveness for this cylinder part as follows:

- Scenario One – single parameter of dimension changed (cylinder height), single rule applied (cylinder rule 01),
- Scenario Two – single parameter of dimension changed (cylinder diameter), single rule applied (cylinder rule 02).

These scenarios will be tested later in the following section.

6.3.3.2 Virtual Product Modelling Framework Application

The next step is to utilise the cylinder part data as the input to evaluate the virtual product modelling framework. The following sections focus on each step of the proposed virtual product modelling framework in application to this cylinder part.

a) Product model development

The cylinder part can also be represented by using the VPM product model structure to show a clear structure of atomic decomposition of product knowledge (shown in Figure 6-23).

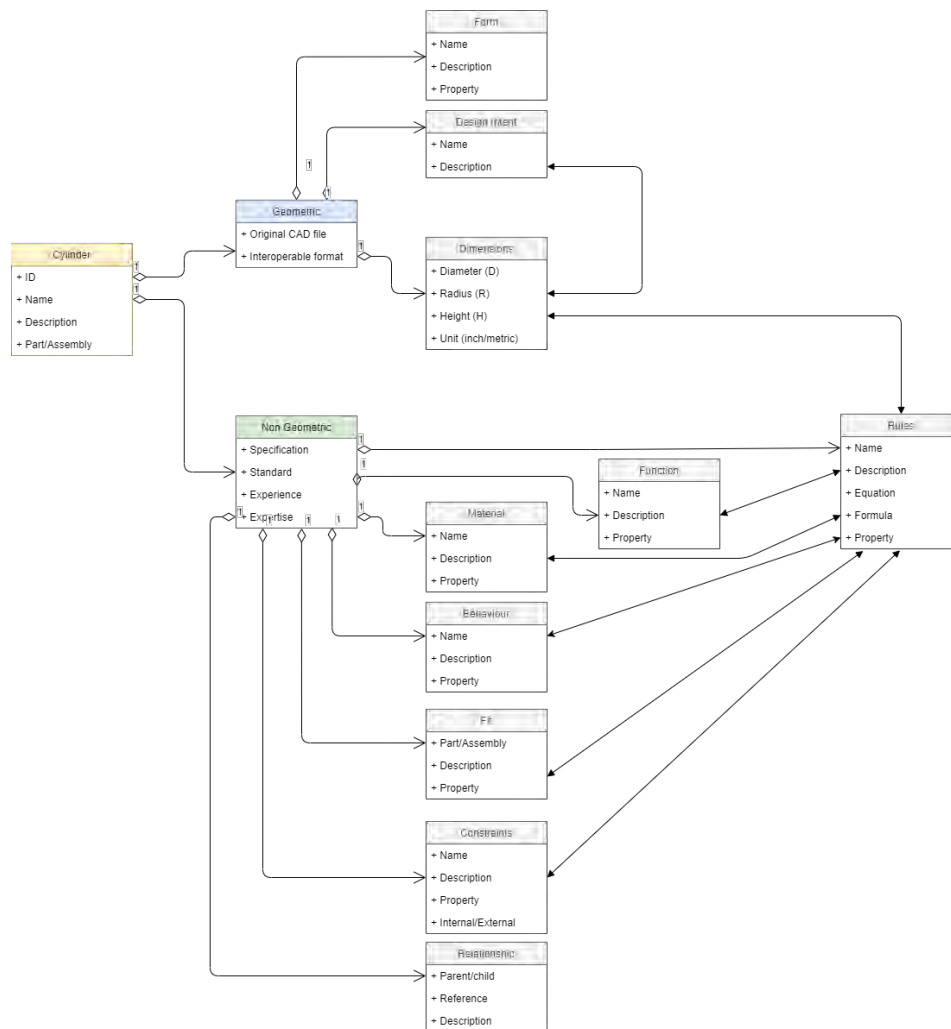


Figure 6-23: VPM product model structure of the cylinder part in UML diagram

b) Knowledge capture of non-geometric information

In this step, all the existing information is captured through the use of the knowledge capture tool. The existing information is input into the knowledge capture tool, and then a knowledge file (.xml) that stores all the captured non-geometric information is created (as shown in Figure 6-24).

```
<knowledge>
  <product type="part">
    <id>00000002</id>
    <name>Cylinder</name>
    <description>Primitive Design Feature</description>
    <design_intent>
      <name>Primitive design feature</name>
      <description>To create other design features</description>
    </design_intent>
    <function_>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </function_>
    <form>
      <name>Cylinder</name>
      <description>Geometry: Cylinder</description>
    </form>
    <material>
      <name>not defined</name>
      <description>None</description>
      <property>None</property>
    </material>
    <behaviour>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </behaviour>
    <fit>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </fit>
    <relationship>
      <parent>None</parent>
      <children>None</children>
      <reference>None</reference>
      <description>None</description>
    </relationship>
    <rules>
      <name>Cylinder_rule01</name>
      <description>The height of cylinder should not be larger than 200 </description>
      <formula> </formula>
      <equation> </equation>
    </rules>
    <rules>
      <name>Cylinder_rule02</name>
      <description>The diameter of cylinder should not be larger than 80 </description>
      <formula> </formula>
      <equation> </equation>
    </rules>
  </product>
</knowledge>
```

Figure 6-24: Example - part of knowledge file of the cylinder part

The cylinder part can be further represented with the expanded knowledge using VPM in the UML diagram (shown in Figure 6-25).

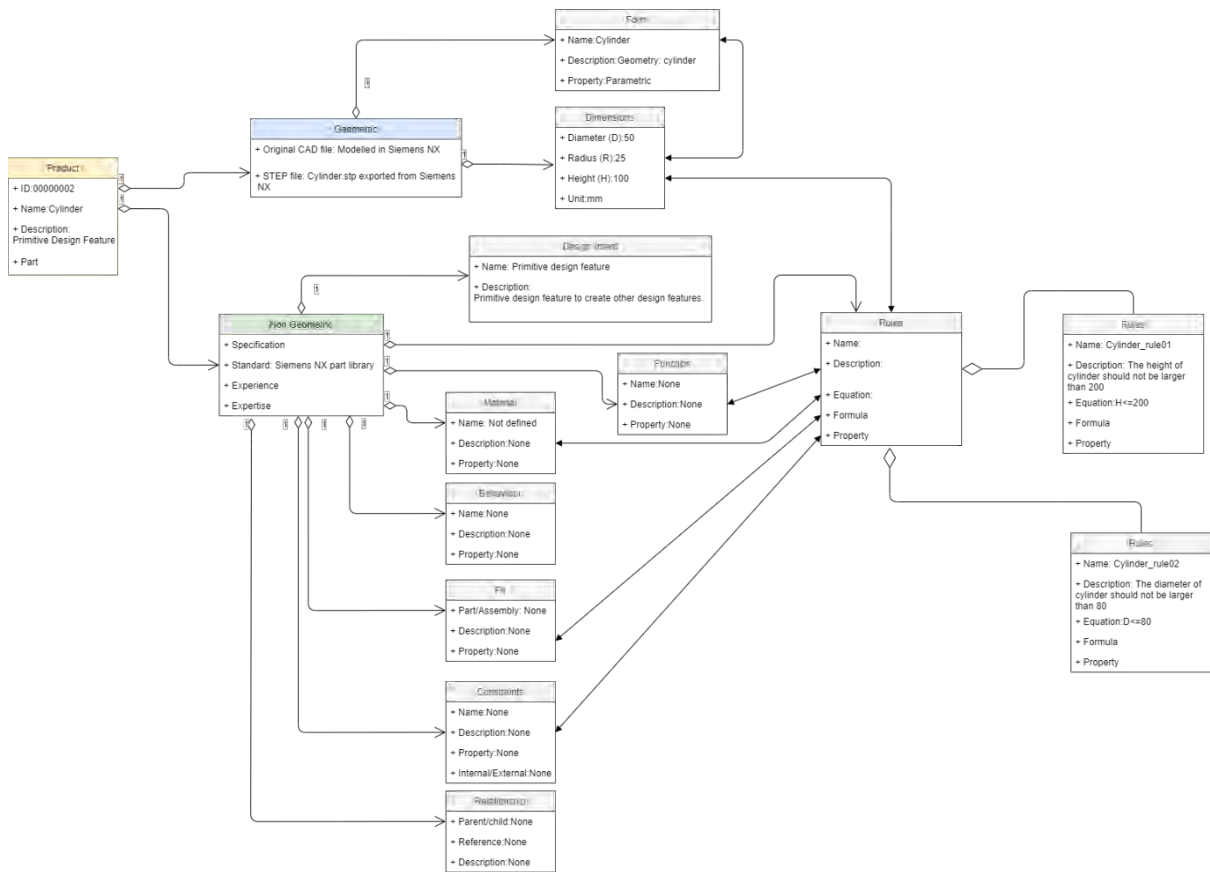


Figure 6-25: VPM product model structure of the cylinder part with captured knowledge

c) Knowledge capture of geometry information

The process of extracting the geometry information of the cylinder remains the same as explained in Section 6.3.2.2 (c). The pre-modelled cylinder part in Siemens NX 10 is shown in Figure 6-26.

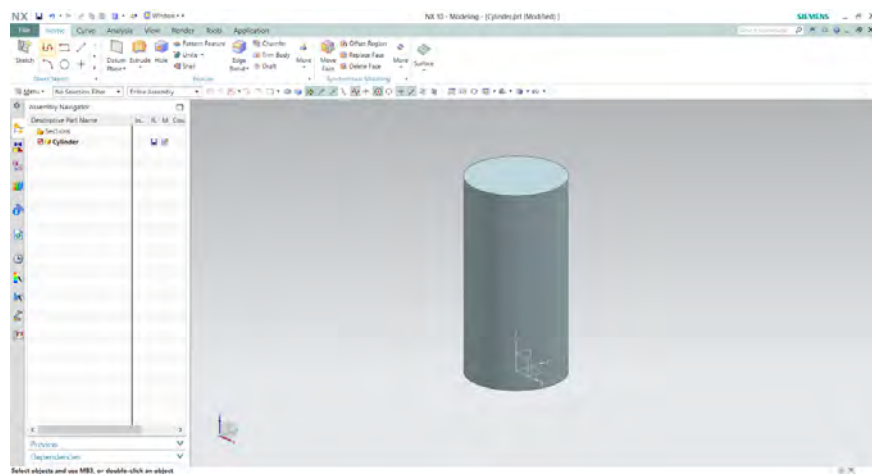


Figure 6-26: A cylinder model created in Siemens NX 10

d) Knowledge mapping

For this cylinder part, two rules are captured and stored in the knowledge file, which are:

- Rule 01: The height of the cylinder should not be larger than 200 ($h \leq 200$)
- Rule 02: The diameter of the cylinder should not be larger than 80.

These rules are further converted into programming logic that constrains the cylinder dimension parameters. Thus, when the user is trying to change the height of the cylinder in the interface, the algorithm will run and tell the user that the changes are affected by the defined rule 01. Similarly, if the diameter of the cylinder is changed, the algorithm will show that the change of diameter is affected by the defined rule 02. Figure 6-27 illustrates the mapping process between the knowledge file and the user interface for visualisation. The full codification is provided in Appendix.

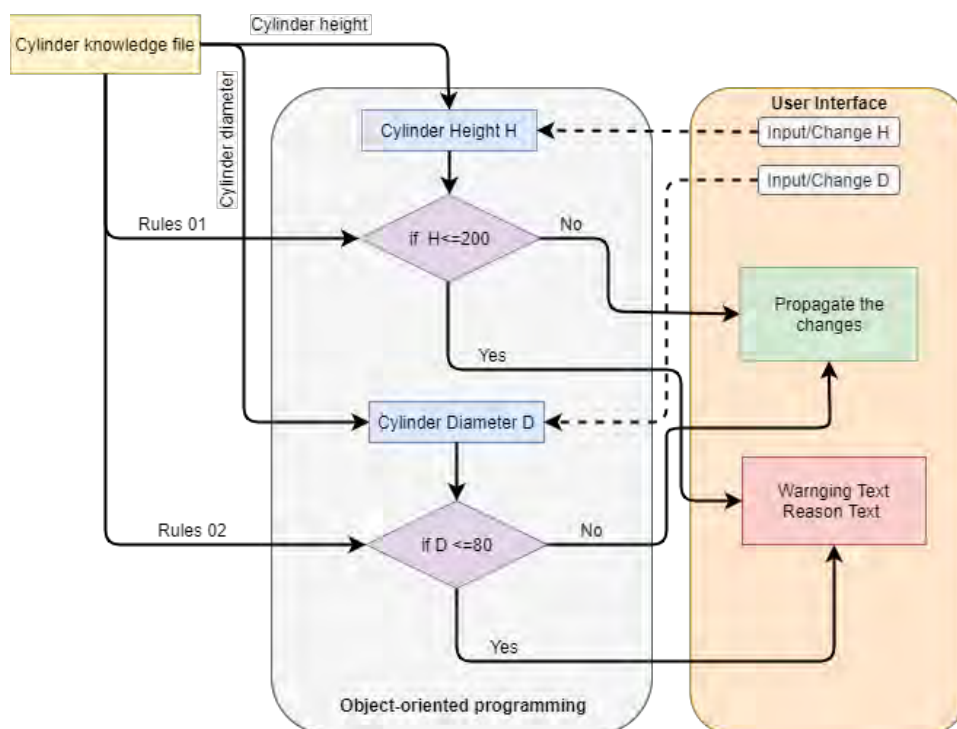


Figure 6-27: Illustration of the knowledge mapping for the cylinder part in use case 1

e) Visualisation & validation

The visualisation is performed in the same way as explained in Section 6.3.2.2 (e) (as shown in Figure 6-28).

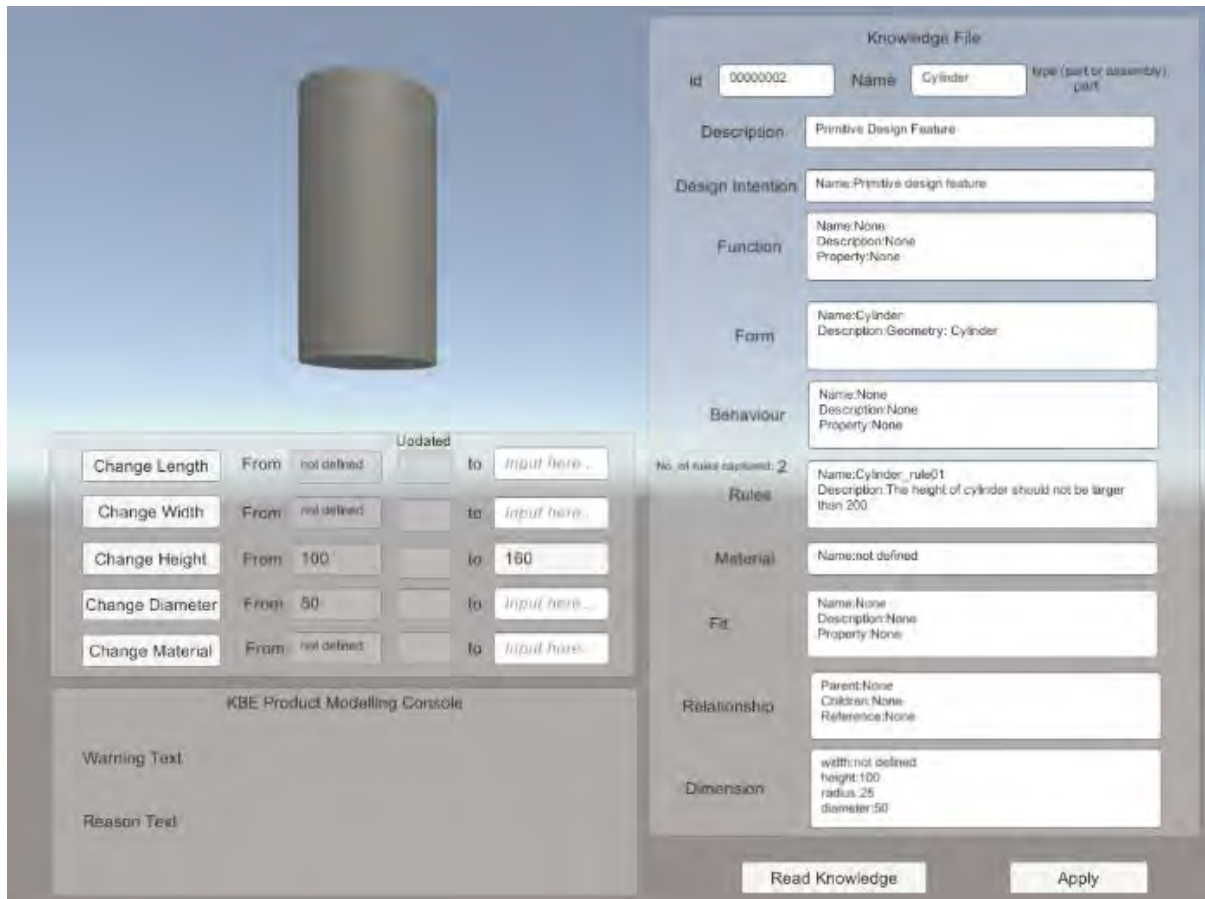
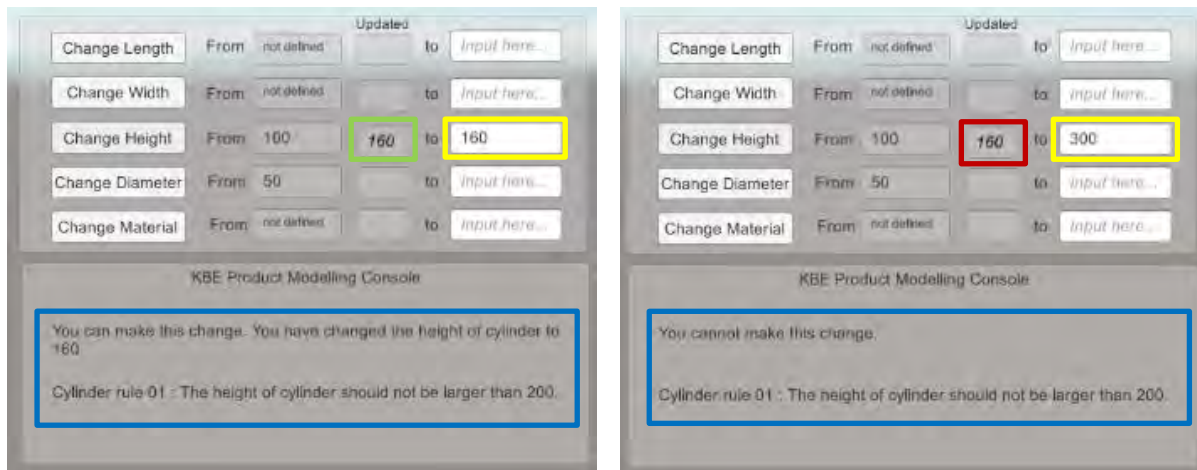


Figure 6-28: Cylinder part model visualised in the developed knowledge-based product modelling environment

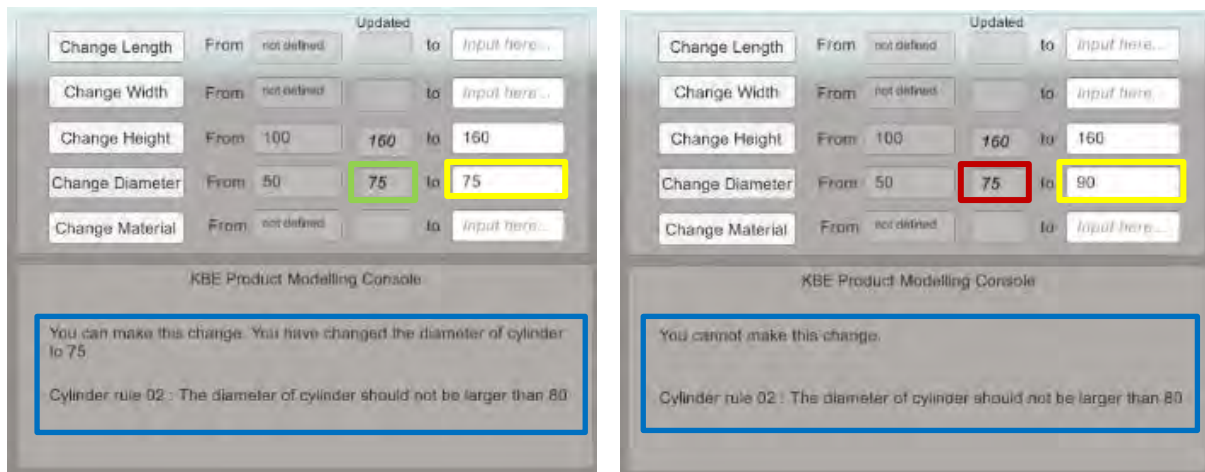
Different from the previous block part, two testing scenarios are identified to test the tool's effectiveness of changing two parameters and applying two rules in one part, which are:

- Scenario One - single parameter of dimension changed (cylinder height), single rule applied (cylinder rule 01),
- Scenario Two - single parameter of dimension changed (cylinder diameter), single rule applied (cylinder rule 02).

After the users select to change the cylinder height by clicking the button “Change Height” in the tool interface, the changes will be propagated in the “KBE product modelling console” panel. The results of the two testing scenarios are shown in Figure 6-29. In the meantime, the “KBE product modelling console” also shows the rule affecting the changes. These results will be further discussed and analysed at the end of this use case in Section 6.3.6.



(a) Scenario One - change of height with rule 01



(b) Scenario Two - change of diameter with rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Red box - propagated parameter (changes not allowed by rules); Blue box - knowledge reasoning.

Figure 6-29: Results of validation – use case 1: cylinder part

6.3.4 Simple Part – a Cone Part

Following the same process explained previously in Section 6.3.2, the existing information of a simple cone part is collected and shown in Table 6-5. The listed product information is assumed to be the non-geometric information that is going to be captured for virtual product modelling.

Table 6-5: Existing information of use case 1- primitive design feature: cone part

VPM knowledge class	Existing product information
Product	Cone
Feature	Primitive design feature - cone
Description	Primitive Design Feature
Function	None
Behaviour	None
Form	Geometry: cone. A circular base and one continuous curve.
Material	Not defined
Design intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Base Diameter =10 mm, Height = 18mm.
Rules	Rule 01: The base diameter of cone should be among 10,16,18,20 mm Rule 02: If the diameter of cone is less than 16mm, the height should be 18mm. If the diameter of cone is equal to or larger than 16 mm, the height should be 24 mm.
Fit	None
Constraint	None
Relationship	None
Reference	None

6.3.4.1 Measurement Parameters and Testing Scenarios

Measurement parameters defined for the first use case evaluation are utilised for this cone part (as shown in Table 6-3). The following two testing scenarios are identified for this cone part to verify and validate the tool's effectiveness:

- Scenario One - single parameter of dimension changed (cone base diameter), single rule applied (cone rule 01),
- Scenario Two - single parameter of dimension changed (cone height), single rule applied (cone rule 02).

These scenarios will be tested later in the following section.

6.3.4.2 Virtual Product Modelling Framework Application

The next step is to utilise the cone part data as the input to evaluate the virtual product modelling framework. The following sections explain each step of the proposed virtual product modelling framework in application to this cone part.

a) Product model development

Similar to the previous two simple parts, the cone part can be represented by using the VPM product model structure to provide a clear structure of atomic decomposition of product knowledge (shown in the following Figure 6-30).

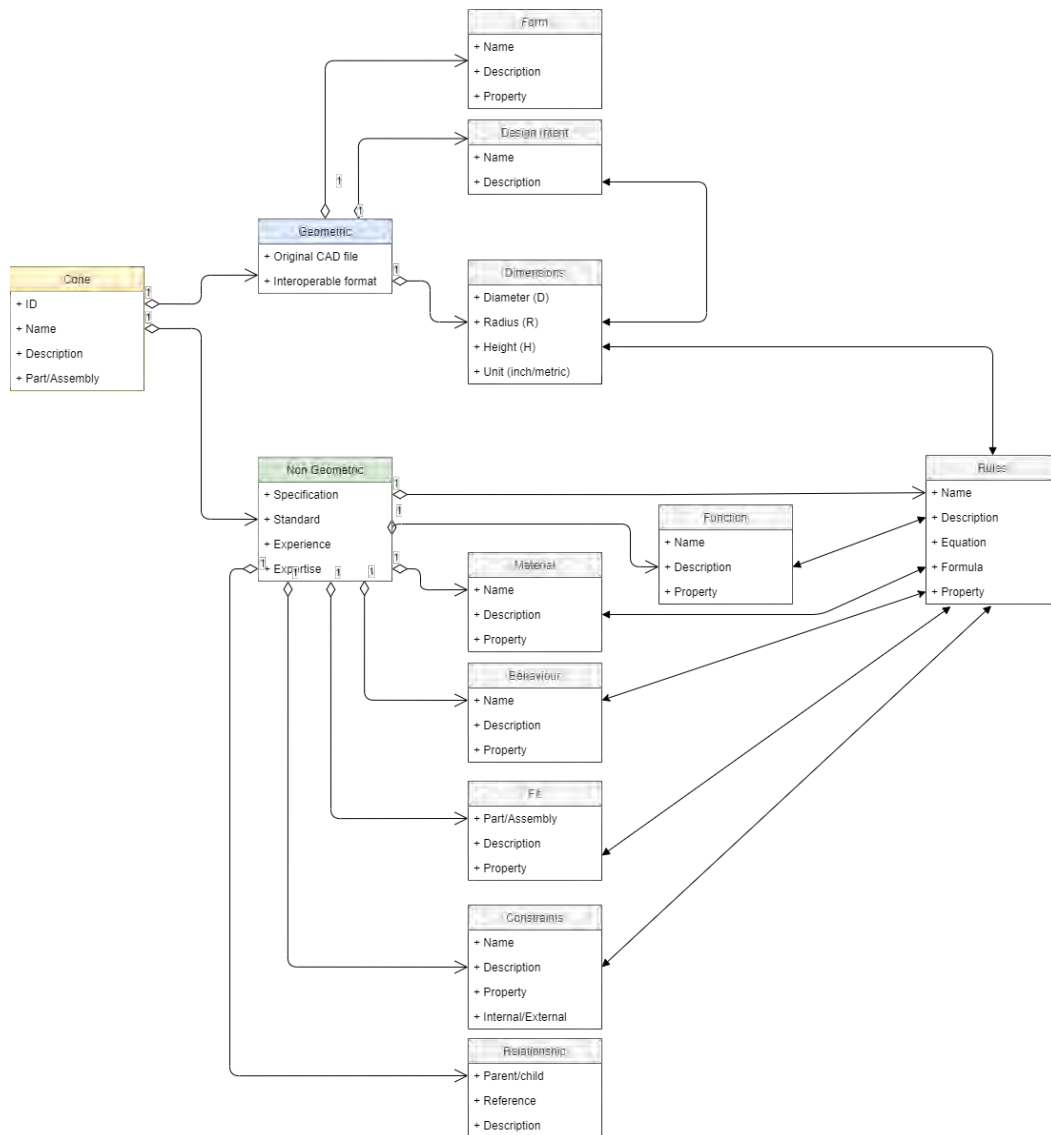


Figure 6-30: VPM product model structure of the cone part in UML diagram

b) Knowledge capture of non-geometric information

The process of capturing the cone's existing non-geometric information is the same as has been explained in Section 6.3.2.2 (b). The generated knowledge file from the knowledge capture tool is shown in Figure 6-31.

```

<knowledge>
  <product type="part">
    <id>00000004</id>
    <name>Cone</name>
    <description>Primitive Design Feature</description>

    <design_intent>
      <name>Primitive design feature</name>
      <description>To create other design features</description>
    </design_intent>
    <function_>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </function_>

    <form>
      <name>Cone</name>
      <description>Geometry: cone. A circular base and one continuous curve.</description>
    </form>

    <material>
      <name>not defined</name>
      <description>None</description>
      <property>None</property>
    </material>

    <behaviour>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </behaviour>

    <fit>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </fit>

    <relationship>
      <parent>None</parent>
      <children>None</children>
      <reference>None</reference>
      <description>None</description>
    </relationship>

    <rules>
      <name>Cone_rule01</name>
      <description>The base diameter of cone should be among 10,16,18,20 mm </description>
      <formula> </formula>
      <equation> </equation>
    </rules>
    <rules>
      <name>Cone_rule02</name>
      <description>If the diameter of cone is less than 16mm, the height should be 18mm.
      If the diameter of cone is equal to or larger than 16 mm, the height should be 24 mm.</description>
      <formula></formula>
      <equation></equation>
    </rules>
  </product>
</knowledge>

```

Figure 6-31: Example - part of knowledge file of the cone part

The cone part can be further represented with the captured knowledge through VPM in the UML diagram (shown in Figure 6-32).

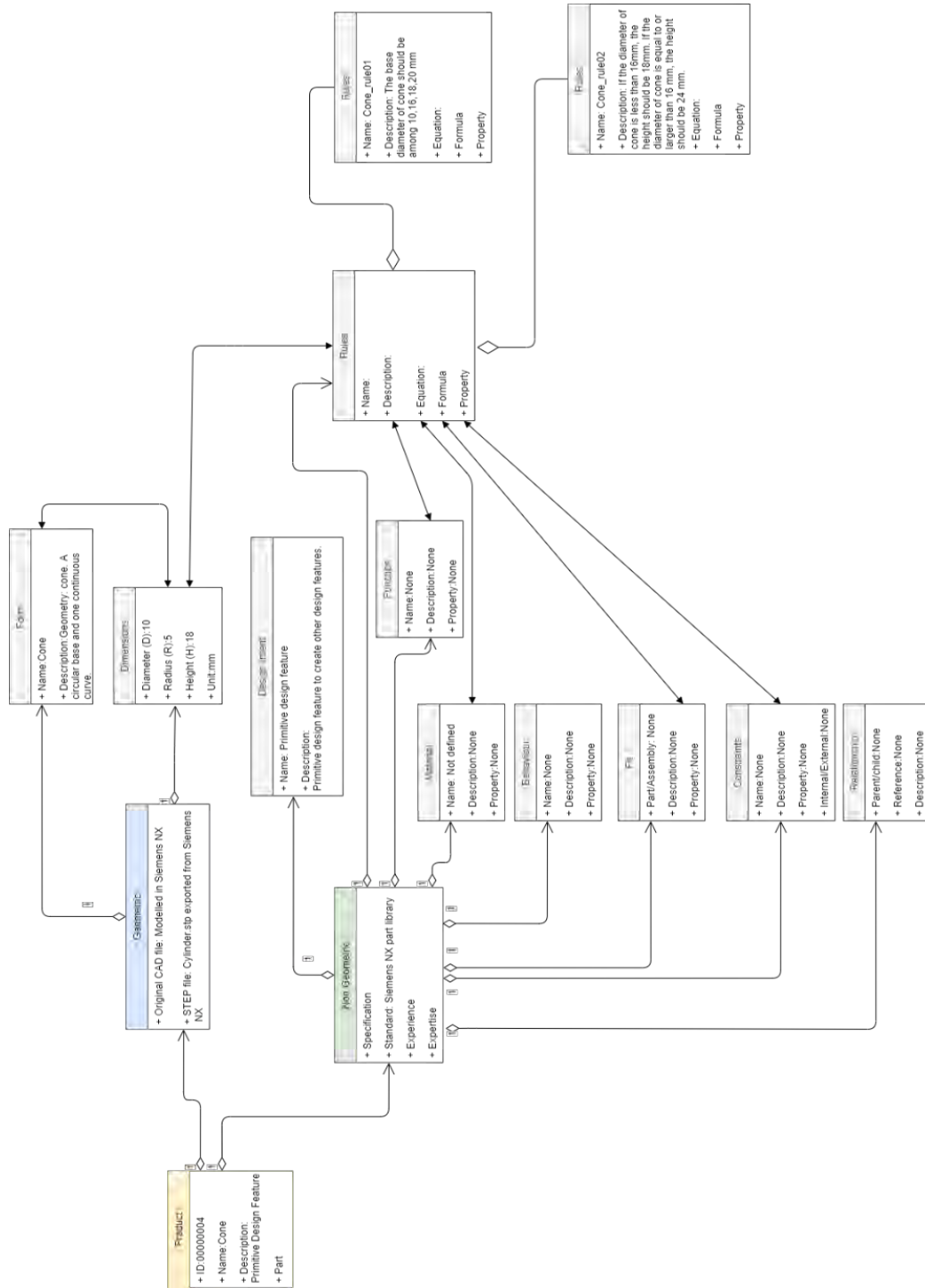


Figure 6-32: VPM product model structure of the cone part with captured knowledge

c) Knowledge capture of geometry information

Extracting the geometry information of the cone part follows the same process explained in Section 6.3.2.2 (c). The pre-modelled cone part in Siemens NX 10 is shown in Figure 6-33.

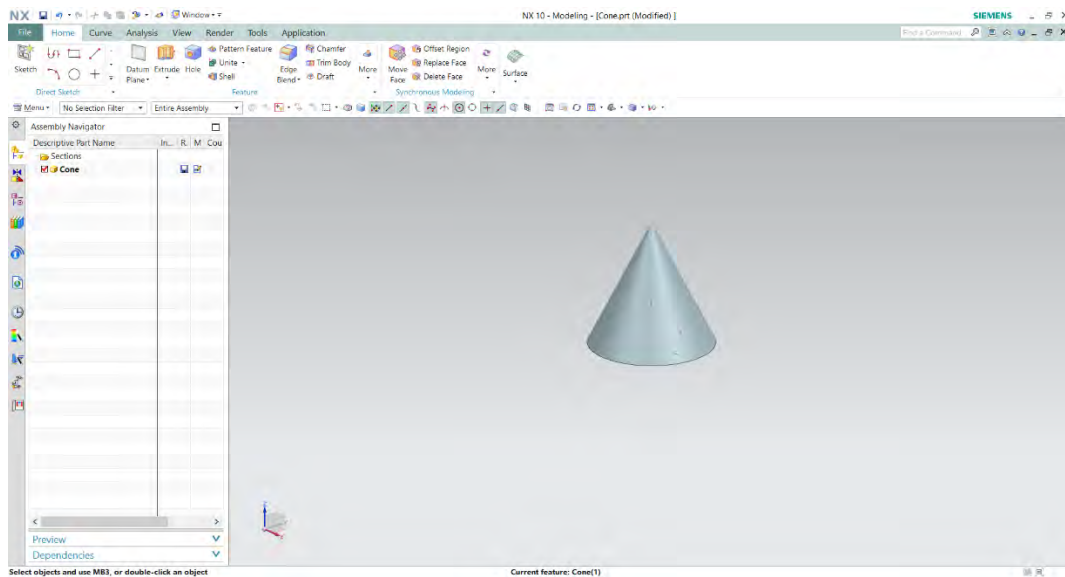


Figure 6-33: A cone model created in Siemens NX 10

d) Knowledge mapping

For this cone part, two rules are captured and stored in the knowledge file, which are:

- Rule 01: The base diameter of the cone should be 10,16,18,20 mm
- Rule 02: If the base diameter of the cone is less than 16mm, the height should be 18mm. If the base diameter of the cone is equal to or larger than 16 mm, the height should be 24 mm.

These rules are further converted into object-oriented programming logic that constrains the cone dimension parameters. Thus, when the user is trying to change the base diameter of the cone in the interface, the algorithm will run and tell the user that the changes are affected by the defined cone rule 01. If the base diameter of the cone is being changed to less than 16 mm, the algorithm will show that the height of the cone should be 18mm because of the defined rule 02. Similarly, if the base diameter of the cone is being changed to equal to or larger than 16 mm, the algorithm will tell that the height of the cone should be 24 mm under the constraints of cone rule 02.

Another test is to change the height of the cone. With the constraint of cone rule 02, the height of the cone is dominated by the base diameter. Therefore, when the user gives a height value in the interface, the algorithm will run and check if the current diameter is less than, equal to, or larger than 16mm. And the allowed height value would only be 18mm or 24 mm. Figure 6-34 describes the mapping process between the knowledge file and the user interface for visualisation. The detailed codification is provided in Appendix.

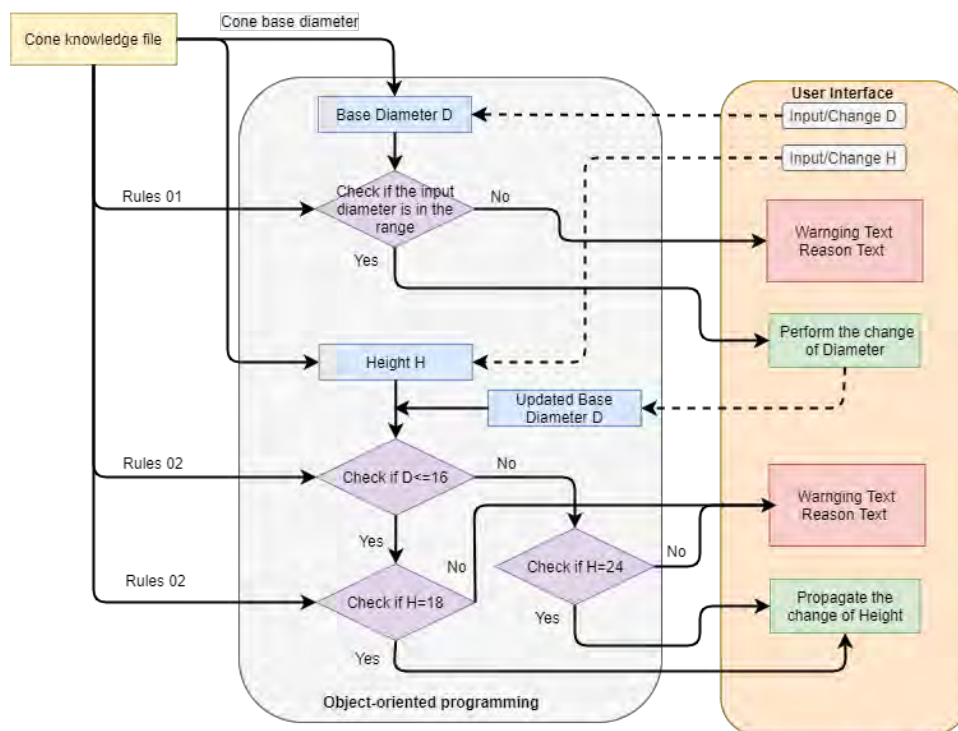


Figure 6-34: Illustration of the knowledge mapping for the cone part in use case 1

e) Visualisation & validation

The visualisation of this cone part (as shown in Figure 6-35) is performed in the same way as been explained in Section 6.3.2.2 (e).

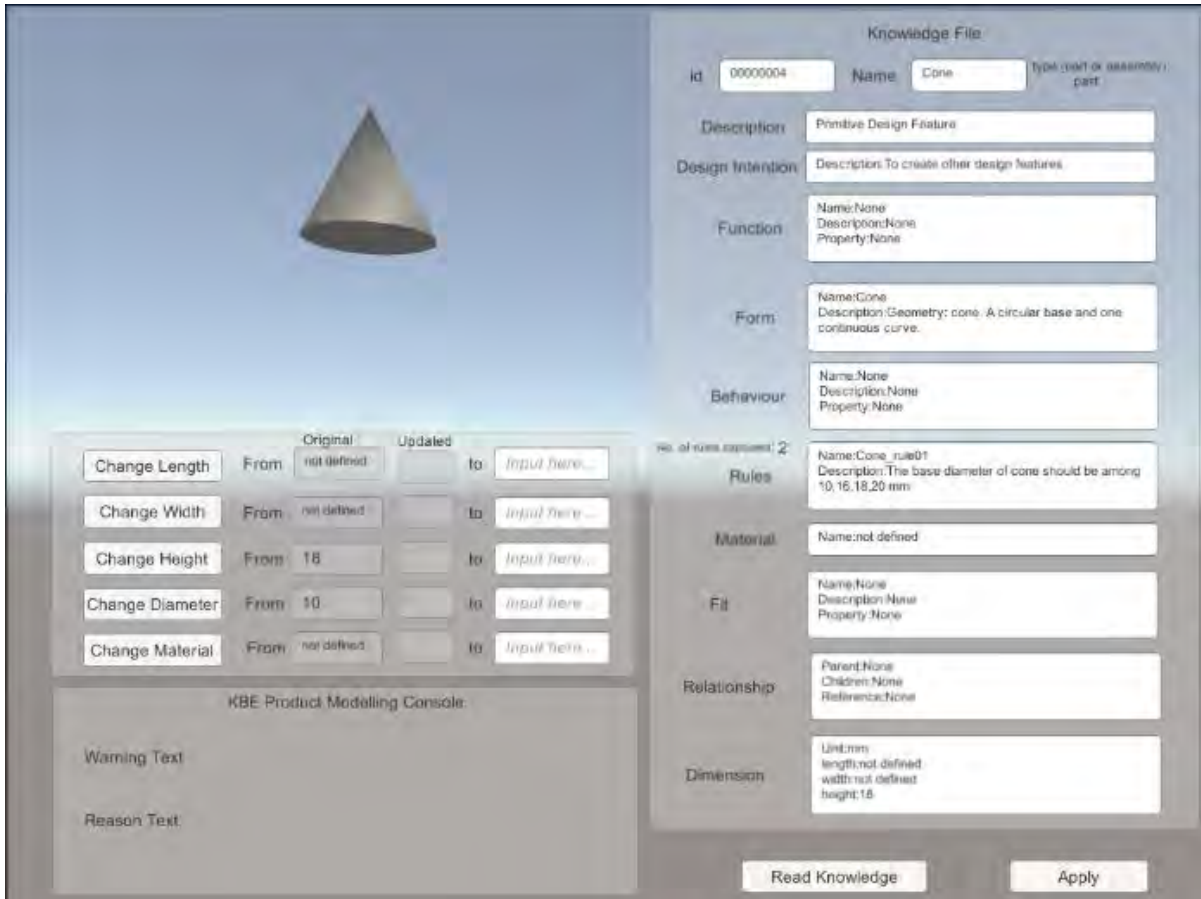


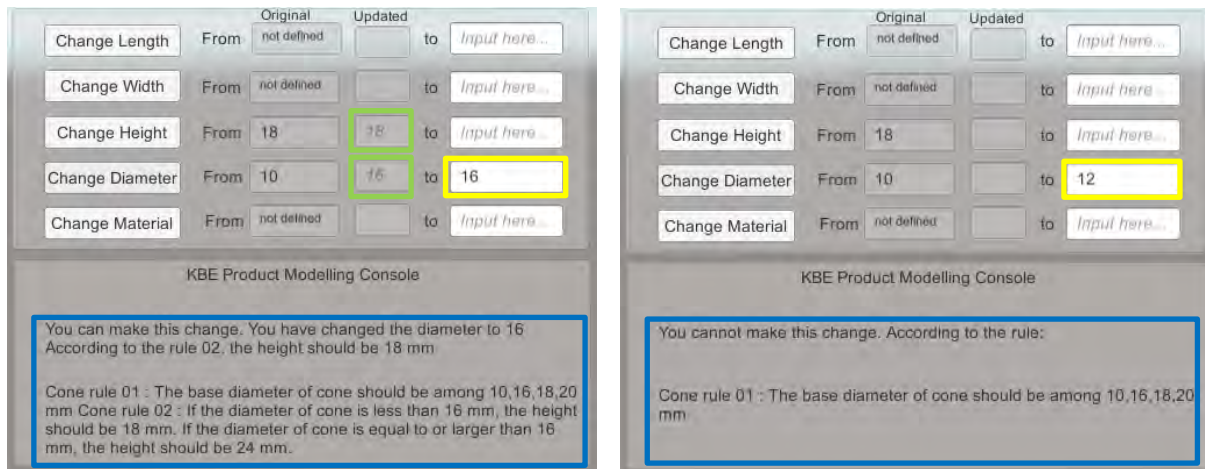
Figure 6-35: Cone part model visualised in the developed knowledge-based product modelling environment

Two testing scenarios have been identified before as follows:

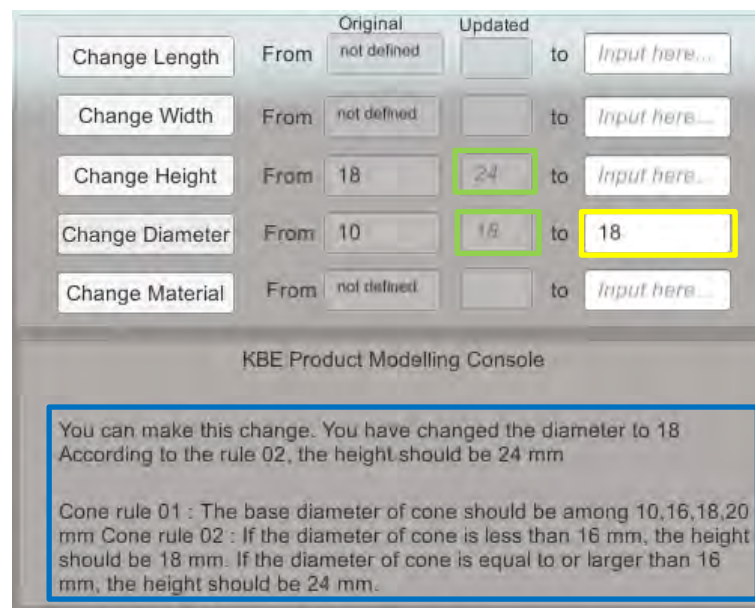
- Scenario One - single parameter of dimension changed (cone base diameter), single rule applied (cone rule 01)
- Scenario Two - single parameter of dimension changed (cone height), single rule applied (cone rule 02).

After the user selects to change the cone base diameter by clicking the “Change Diameter” button in the tool interface, the changes will be reflected in the “KBE product modelling console” panel. Similar, the user can change the height of the cone by clicking “Change Height” in the tool interface. The results of the two testing scenarios are shown in Figure 6-

36. These results will be further discussed and analysed at the end of this use case in Section 6.3.6.



(a) Scenario One - change of base diameter



(b) Scenario One and Two - propagation of the change of base diameter to the change of height

	Original	Updated
Change Length	From not defined	to <input type="text" value="Input here..."/>
Change Width	From not defined	to <input type="text" value="Input here..."/>
Change Height	From 18	to <input type="text" value="30"/>
Change Diameter	From 10	to <input type="text" value="Input here..."/>
Change Material	From not defined	to <input type="text" value="Input here..."/>

KBE Product Modelling Console

The height of this cone is dominated by the diameter
According to the following rules:

Cone rule 02 : If the diameter of cone is less than 16 mm, the height should be 18 mm. If the diameter of cone is equal to or larger than 16 mm, the height should be 24 mm.

(c) Scenario Two - change of height against cone rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-36: Results of validation – use case 1: cone part

6.3.5 Simple Part - a Sphere Part

The existing information of a sphere part is collected from the part library of Siemens NX 10 by performing the same process as described in Section 6.3.2. A list of information on the sphere part that will be captured for virtual product modelling is shown in the following Table 6-6.

Table 6-6: Existing information of use case 1- primitive design feature: sphere part

VPM knowledge class	Existing product information
Product	Sphere
Feature	Primitive design feature - Sphere
Description	Primitive Design Feature
Function	None
Behaviour	None
Form	Geometry: Sphere. Could be solid or hollow
Material	304 Stainless Steel
Design Intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Diameter =25 mm
Rules	<p>Rule 01: The diameter of sphere should be among 19,20,21,22,25,30,35,40 mm</p> <p>Rule 02: The material of the steel ball should be among American Iron and Steel Institute standard (AISI) 201, AISI 304, AISI 316 stainless steel</p>
Fit	None
Constraint	None
Relationship	None
Reference	None

6.3.5.1 Measurement Parameters and Testing Scenarios

Measurement parameters for this sphere part are the same as described in Table 6-3. The following two testing scenarios are identified for this sphere part to verify and validate the tool's effectiveness:

- Scenario One - single parameter of dimension changed (sphere diameter), single rule applied (sphere rule 01)
- Scenario Two - single parameter of dimension changed (sphere material), single rule applied (sphere rule 02)

These scenarios will be tested later in the following section.

6.3.5.2 Virtual Product Modelling Framework Application

Next, the sphere part data is used as the input to evaluate the virtual product modelling framework. The following sections explain each stage of the proposed virtual product modelling framework in application to this sphere part.

a) Virtual product model development

Like the previous simple part, the sphere part can also be represented by using the VPM product model structure (shown in Figure 6-37).

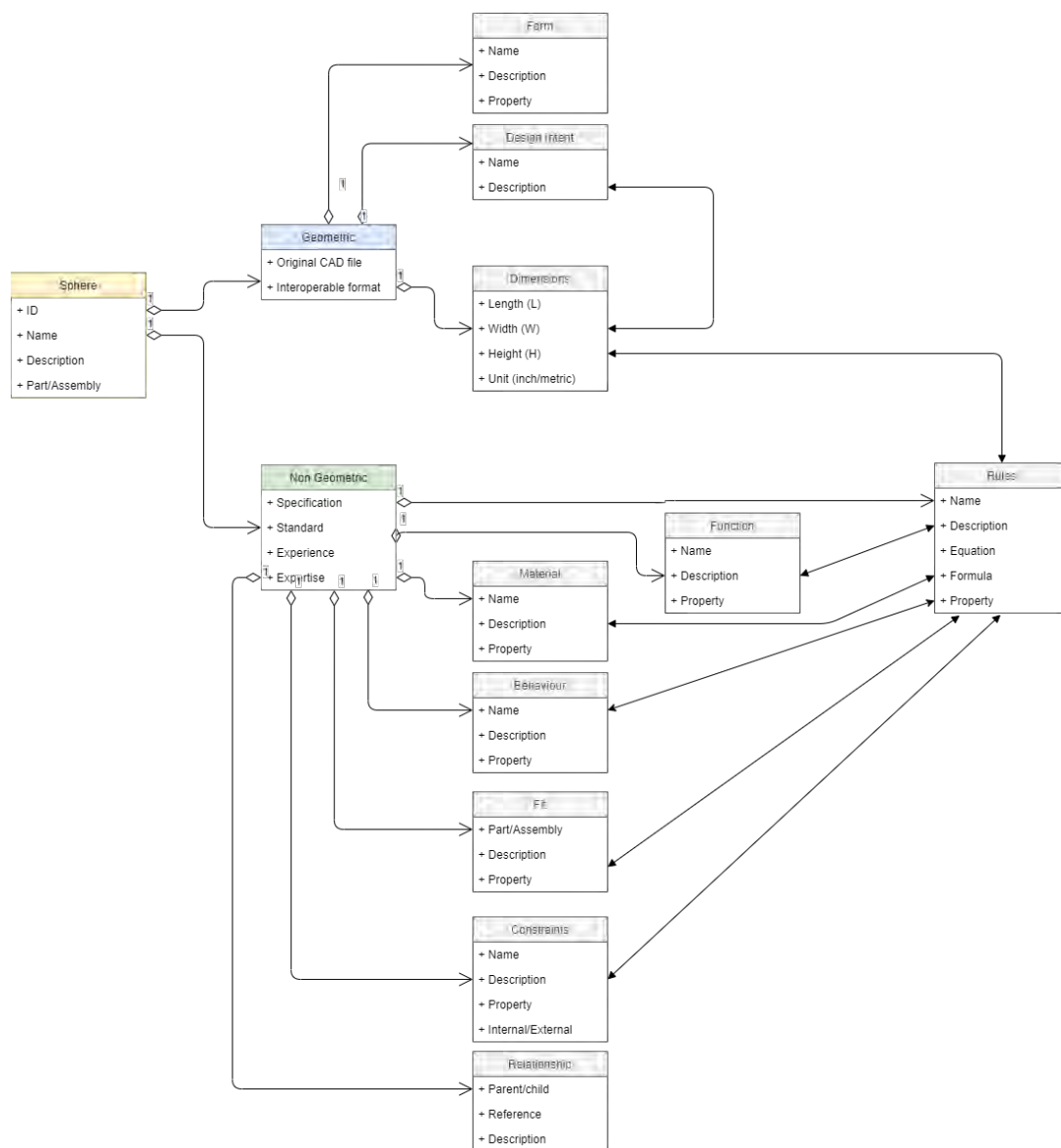


Figure 6-37: VPM product model structure of the sphere part in UML diagram

b) Knowledge capture of non-geometric information

The process of capturing the sphere part's existing non-geometric information is the same as explained in Section 6.3.2.1 (b). The knowledge file of the sphere created by the knowledge capture tool is shown in Figure 6-38.

```
<knowledge>
  <product type="part">
    <id>00000003</id>
    <name>Sphere</name>
    <description>Primitive Design Feature</description>
    <design_intent>
      <name>Primitive design feature</name>
      <description>To create other design features</description>
    </design_intent>
    <function_>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </function_>
    <form>
      <name>Sphere</name>
      <description>Geometry: Sphere. Could be solid or hollow</description>
    </form>
    <material>
      <name>304 Stainless Steel</name>
      <description>The sphere is made of 304 stainless steel. </description>
      <property>None</property>
    </material>
    <behaviour>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </behaviour>
    <fit>
      <name>None</name>
      <description>None</description>
      <property>None</property>
    </fit>
    <relationship>
      <parent>None</parent>
      <children>None</children>
      <reference>None</reference>
      <description>None</description>
    </relationship>
    <rules>
      <name>Sphere_rule01</name>
      <description>The diameter of sphere should be among 19,20,21,22,25,30,35,40 mm </description>
      <formula> </formula>
      <equation> </equation>
    </rules>
    <rules>
      <name>Sphere_rule02</name>
      <description>The material of the steel ball should be among AISI 201, AISI 304, AISI 316 stainless steel</description>
      <formula></formula>
      <equation></equation>
    </rules>
  </product>
</knowledge>
```

Figure 6-38: Example - part of knowledge file of the sphere part

The sphere part can be further represented with the captured knowledge in VPM using the UML diagram (as shown in Figure 6-39).

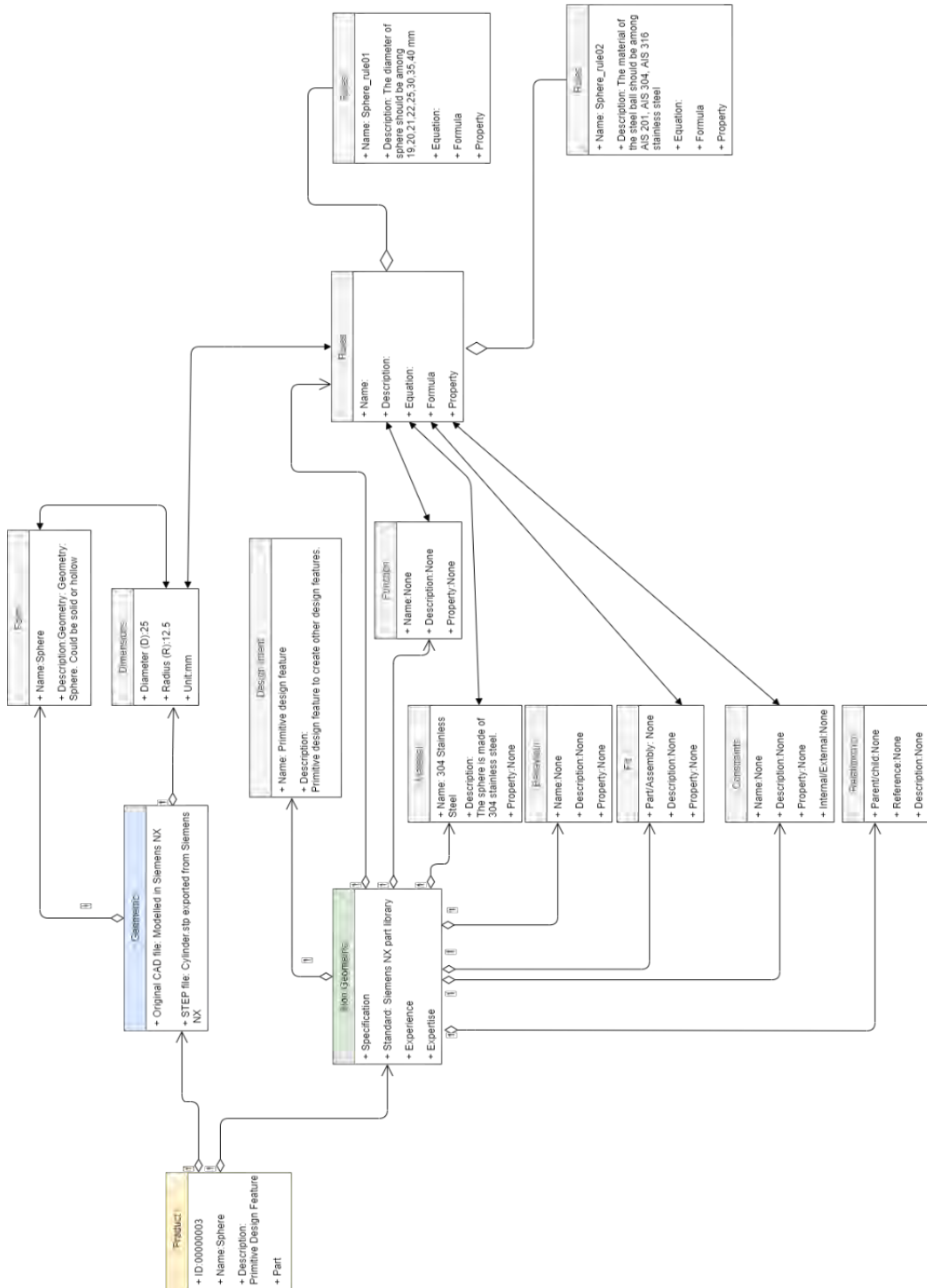


Figure 6-39: VPM product model structure of the sphere part with captured knowledge

c) Knowledge capture of geometry information

The process of extracting the geometry information of the sphere continues as same as been explained in Section 6.3.2.2 (c). The sphere is pre-modelled in Siemens NX 10 as shown in Figure 6-40.

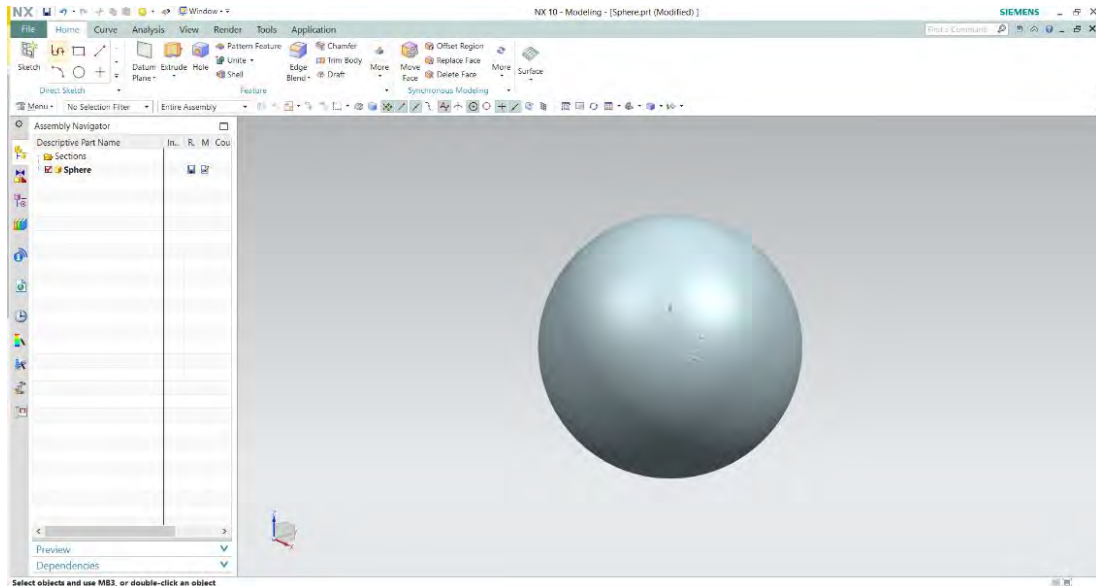


Figure 6-40: A sphere model created in Siemens NX 10

d) Knowledge mapping

For this sphere part, two rules are captured and stored in the knowledge file, which are:

- Rule 01: The diameter of sphere should be among 19,20,21,22,25,30,35,40 mm
- Rule 02: The material of the steel ball should be among AISI 201, AISI 304, and AISI 316 stainless steel.

These rules are further converted into programming logic constraining the sphere diameter and material. Therefore, when the user inputs a new value of diameter and presses the “Change Diameter” button in the interface, the unique algorithm will run and check with the rule01 and then tell the user whether the change could be executed along with the reason. If the user inputs a new material and presses the “Change Material” button, the algorithm will examine if this material is applicable for the sphere based on rule 02. Figure 6-41 shows the

mapping process between the knowledge file and the user interface for visualisation through object-oriented programming. The complete coding is attached in Appendix.

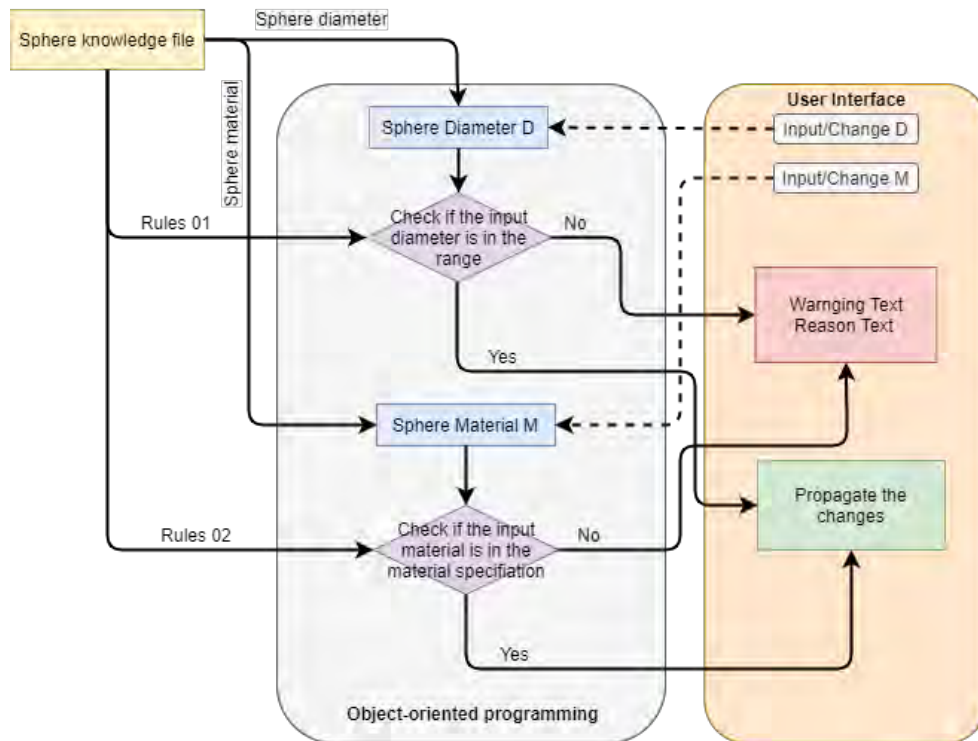


Figure 6-41: Illustration of the knowledge mapping for the sphere part in use case 1

e) Visualisation & validation

The visualisation of this sphere part is carried out in the same way as described in Section 6.3.2.2 (e). The visualisation result is shown in text and 3D visualisation (see Figure 6-42).

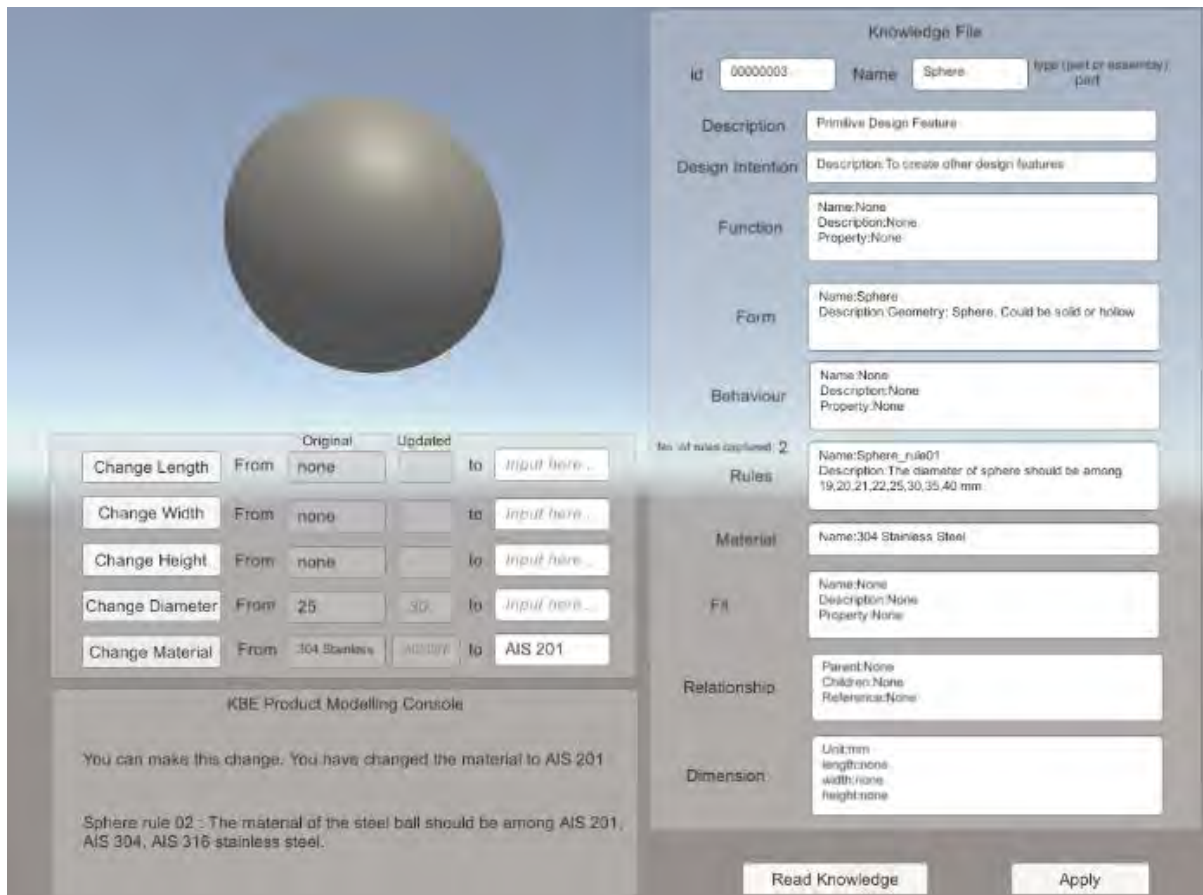
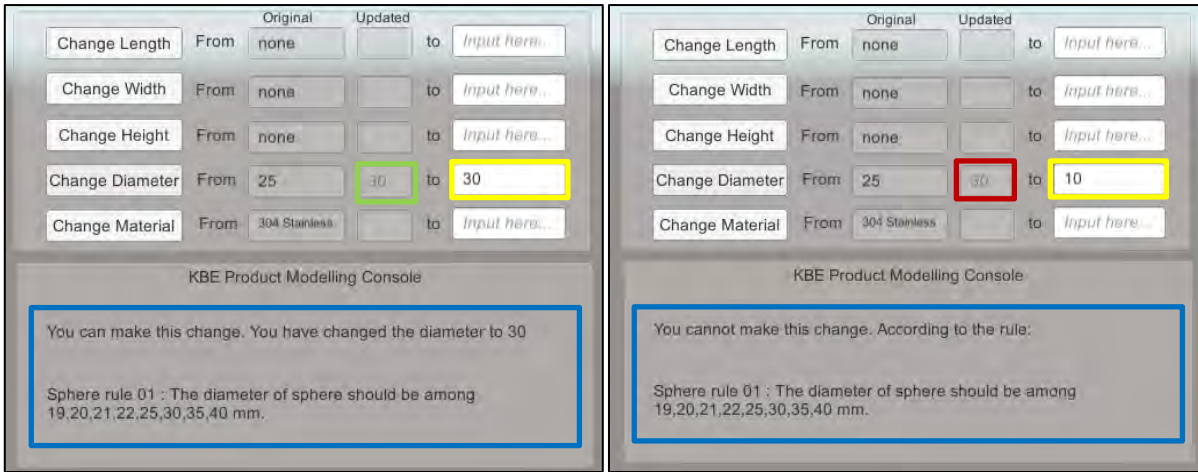


Figure 6-42: Sphere part model visualised in the developed knowledge-based product modelling environment

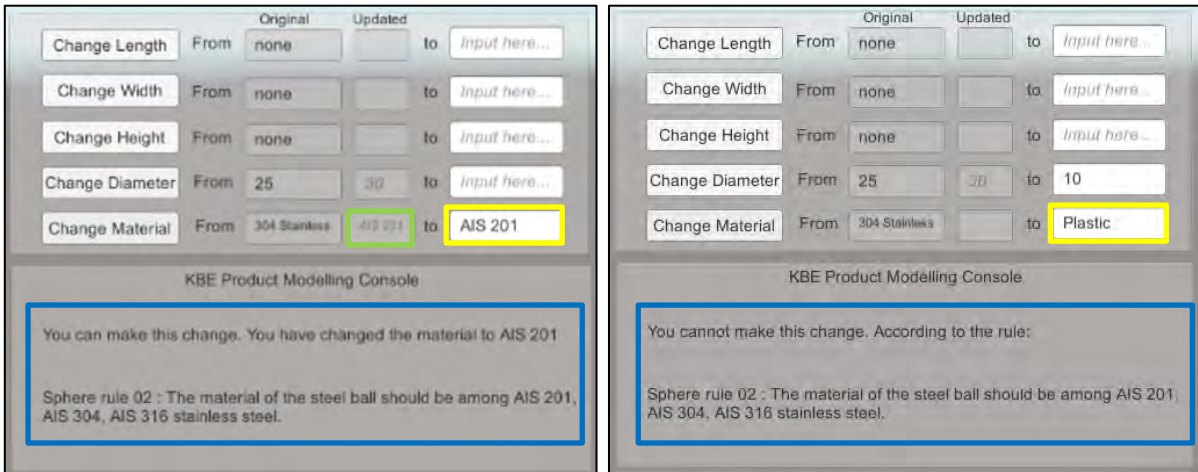
Two testing scenarios are identified in Section 6.3.5.1 to test the tool effectiveness of changing single parameters and applying rules in one part, which are:

- Scenario One - single parameter of dimension changed (sphere diameter), single rule applied (sphere rule 01),
- Scenario Two - single parameter of dimension changed (sphere material), single rule applied (sphere rule 02).

The results of the two testing scenarios are shown in Figure 6-43 and will be further discussed and analysed at the end of use case 1 in the following section.



(a) Scenario One - change of diameter under rule 01



(b) Scenario Two - change of material under rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Red box - propagated parameter (changes not allowed by rules); Blue box - knowledge reasoning.

Figure 6-43: Results of validation – use case 1: sphere part

6.3.6 Result Analysis and Use Case Discussion

After performing the complete cycle of the virtual product modelling framework methodology application for the simple part with primitive design features use case, the next evaluation objective is to analyse the virtual product modelling framework results critically and then compare the product modelling results from using the virtual product modelling

framework with the use of current existing/legacy product modelling system for same circumstances.

Based on the pre-defined measurement parameters in Table 6-3, it can be seen that the framework satisfies all of these measurement parameters and shows all the expected results in this first use case implementation.

1) Generative representation – C1

A VPM product model structure is developed for each part in this use case 1, to provide a generative representation of the part by using the atomic blocks with the knowledge classes that have been defined in Section 5.2 of Chapter 5. The developed VPM product model structure showcases that it can capture both geometric information of the product and its corresponding design knowledge. And the captured knowledge offers the potential of showing a set of possibilities based on the knowledge hierarchy from predefined products. For example, the knowledge stored in design intent will provide users with information required to analyse why the product is generated.

However, the level of generalisation depends on the richness of product data. In this particular use case, the existing product information and the product itself are simple. Only a few parameters can be varied to provide different product design configurations. But this does not limit the capability of VPM in providing the generative representation for an engineering product. By giving sufficient data as existing design knowledge, it is possible to capture and include the design strategy as “Rules” that are required to generate a particular product from a specification using VPM.

2) Knowledge capture – C2

Firstly, the virtual product modelling methodology successfully captures all the collected existing product information of the four simple parts in this use case 1, including simple rules that constrain one or two parameters, basic descriptions, simple dimensional parameters, etc.

Secondly, a knowledge file in XML format is generated for each simple part storing all the captured product information. Therefore, through the implementation of this virtual product modelling methodology, the existing product design knowledge is captured, classified, and stored in an exchangeable format. This allows the captured knowledge to be reused for providing supporting data for less-experienced design engineers to understand the product design. Moreover, the developed product model in VPM and the knowledge file can serve as a knowledge reservoir that is accessible to users to provide available product data for other activities at different product development phases.

3) Product geometry and knowledge visualisation – C3

The geometry of the four simple parts in use case 1 and the captured knowledge stored in the knowledge files are visualised successfully in the interface. It proves that the developed knowledge-based product modelling environment can visualise the geometry of these simple parts along with their associated knowledge.

As discussed before in sections 5.1.5 and 5.3.5, visualisation in the developed knowledge-based product modelling environment should provide the ability to view the product model's original geometry and the possible changes that will happen to the geometry and the associated knowledge that constrains the changes. In this use case, the possible changes of dimensions that the users in this interface can make are limited to the use case. And the results of changes in length, width, height, diameter, and material are presented through the text description in the interface. These results help recognise the limitation of the developed knowledge-based product modelling environment in visualisation. The text description is not as effective as a graphical 3D display in visualising geometry. This deficiency is caused by the lack of available enabling technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. However, the limitation in visualisation is acceptable as the system has shown its effectiveness in

propagating the changes of simple parts geometry by reusing the captured knowledge. Visualisation of the original geometry allows the users to know the 3D shapes of the product that they are modelling. The visualisation of knowledge provides them with additional information about each simple part.

4) Product relationship representation – C4

In this use case 1, all the four parts are single parts, and they do not have relationships with other parts. Therefore, to prove the effectiveness and correctness of the relationship representation, all of the four parts are expected to be identified as “part” in the developed knowledge-based product modelling environment before testing. The implementation results verify that the relationship representation has been achieved for simple parts with primitive design features.

5) Knowledge reasoning and reuse – C5

The results of use case 1 prove that, through object-oriented programming, the developed knowledge-based product modelling environment shows the capability of changing parameters of simple parts in context. All the captured rules are implemented and effectively drive and constrain the targeted parameters. The changes of the affected parameters are provided with reasons, and the propagation of affected changes is presented through text indication in the interface.

In this use case 1, as the design parameters for these parts are simple, the resulted knowledge reasoning and reuse in the modelling process are straightforward. For the block part, when the users change the length of the block, the height and width will also be changed according to the rules. It will save the time of making changes to the height and width separately. For the cylinder part, as the rules have constrained the changes of height and diameter, the users will know if the changes that they make are allowed by the rules. This will help them to avoid making mistakes in the product modelling process. Similarly, for the cone part, when the

users change the base diameter, the developed product modelling environment will show if the changes can be made with the reasons. In the meantime, it will also check and apply the change of height as the height of the cone is constrained by the base diameter. This also helps users save time on making changes to different parameters in the product modelling process. For the sphere part, the knowledge reasoning and reuse can help prevent the wrong input of the diameter value. Additionally, for the material of the cone part and sphere part, knowledge reasoning and reuse in the developed product modelling environment helps the users to identify the appropriate material that they can apply to the parts for manufacturing. In this way, it can help avoid errors in other activities of the product development lifecycle, such as product manufacturing, apart from the product modelling process.

6) Correctness of the changes – C6

By comparing the actual implementation results with the expected results from the pre-defined rules, it can be seen that the changes applied to the geometry of the four simple parts through the reuse of the existing knowledge are correct. This states that the developed method will allow effective product modelling in the knowledge-based product modelling environment.

7) Data exchange – C7 and C8

The visualisation of both the geometry and the associated knowledge of the four simple parts also proves that the proposed VPM data exchange method (discussed in the previous sections 5.2.3 and 5.3.2) is successful in communicating with the CAD platform through STEP file and in exchanging existing product information through a knowledge file. This shows that the developed framework will ensure steady data exchange in the knowledge-based product modelling environment.

The above analysis of the results from use case 1 proves that the virtual product modelling methodology satisfies all the measurement parameters. Next, to evaluate the developed VPM

further, it is crucial to compare these implementation results with the modelling results from using the current existing/legacy CAD system for the same circumstances. The comparison is summarised in Table 6-7 based on the identified evaluation criteria from Section 6.2.

Table 6-7: Comparison of the use case 1 implementation results between the existing/legacy CAD system and VPM

Evaluation Criteria	Existing/legacy CAD system implementation	Virtual Product Modelling framework implementation
The capability of generative representation of engineering products in VPM	Use template design feature model as a generative 3D model of a product (geometry representation only and limited to proprietary format). Limited product information is provided.	Develop a VPM product model structure as a generative representation of the product (standardised format - UML) Can provide information, such as design intent and material, and design rules to understand possible product design configurations.
The capability of capturing the product geometry and its associated knowledge from the existing product information	Capture the product geometry through importing/exporting the model into standardised format. Unable to capture the existing associated product knowledge during the product modelling process.	Capture the product geometry through importing the model from CAD systems in a standardised format. Capture existing associated product knowledge during the product modelling process through the knowledge capture tool.
The capability of visualising the product geometry and its associated knowledge	Visualisation of product geometry only	Visualisation of the product's original geometry and its associated knowledge. Text visualisation of the changes

		in geometry parameters.
The capability of presenting every part of the product and the relationships among them	In use case 1, the product is shown as single part in the modelling hierarchy tree and there is no relationship with other parts.	In use case 1, the product is shown as “part” in the proof-of-concept tool interface, and there is no relationship with the other parts.
The capability of propagating changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge	Manual tracking of changes of parameters. The change of product geometry is reflected in 3D visualisation graphically. No knowledge reasoning is provided when changing parameters in the existing CAD systems. No rules are embedded in template model.	Automatic tracking of changes of parameters through the reuse of rules. Text visualisation of the affected parameters in the proof-of-concept tool interface. The change of product geometry is reflected through text description due to the limited available enabling tools. Knowledge reasoning is provided in the proof-of-concept tool interface by reuse of rules from the existing knowledge
The correctness of the changes applied to the product geometry by reuse of the existing knowledge	Manual check-up of the correctness of changes is required.	The changes applied to the product geometry is following the rules from the existing knowledge. The correctness is proved during the validation stage.
The capability and correctness of the product geometry data exchange	The product geometry data is exchanged in the format of a STEP file.	The product geometry data is exchanged in the form of a STEP file. The correctness is

between different platforms		proved during the visualisation process.
The capability and correctness of the knowledge exchange through knowledge file	The existing CAD systems are not able to exchange knowledge between each other using a generalised format.	The knowledge is exchanged in a knowledge file in the format of XML. XML is a platform-independent neutral format for data communication. Knowledge exchange is proved during the implementation and validation stage.

The critical comparison between the product modelling in the existing CAD systems and the proposed virtual product modelling framework implementation further proves the effectiveness of the framework in the chosen simple part use case. However, the difference in modelling simple parts between using VPM and existing/legacy CAD system may not be evident as the existing product information of these simple parts in use case 1 are brief. Moreover, since simple parts are mainly modelled from primitive design features, only a few parameters could be used and varied to create product variants. The design rules involved in the modelling process are limited to constraining a single parameter as the geometry of each part is simple. Therefore, the developed product model using VPM for simple parts would mainly contain geometric data as the knowledge that can be captured and reused for knowledge reasoning is limited. Users may not need design knowledge to be captured and reused as supplementary information to support their modelling process, as understanding and modelling these simple parts is straightforward. In this case, knowledge capture and reuse would not make product modelling using VPM show much distinction with conventional modelling using existing/legacy CAD systems. Although VPM has shown

effectiveness in modelling simple parts in use case 1, modelling simple parts with primitive design features and inadequate knowledge using VPM may not be as efficient as using the existing/legacy CAD systems because these CAD systems offer mature functions in geometric modelling and visualisation. The next use case selects a hex bolt that has more knowledge to evaluate the effectiveness of VPM in application to an engineering part with complex design rules.

6.4 Use Case 2: Basic Engineer Part – Hex Bolt

6.4.1 Use Case Overview

The second use case aims to validate the framework by implementing an engineering component model which has more knowledge. In the second use case, a hex bolt from literature has been chosen as the testing product model as it is one of the most widely used basic engineering parts. Figure 6-44 shows a hex bolt product model from the Siemens NX library.

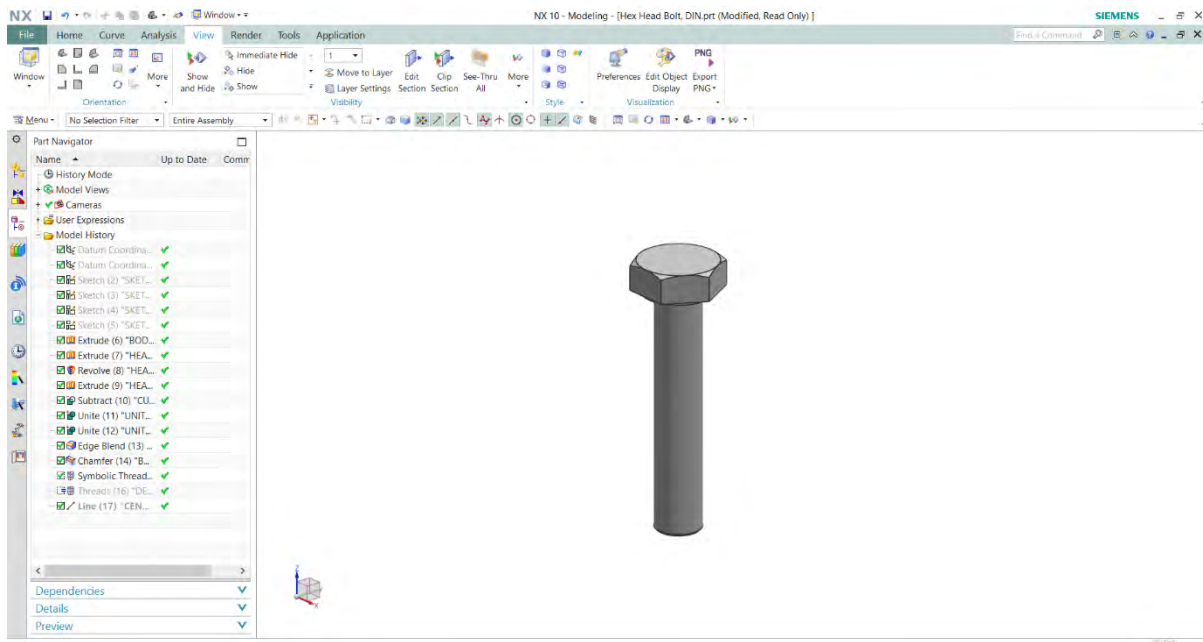


Figure 6-44: A hex bolt modelled in Siemens NX 10

A hex bolt has a hexagonal head and external machine threads for a firm and rough handling. Hex bolts are usually in a wide range of sizes for custom application on their dimensional requirements. The material of hex bolts varies from steel, alloy steel, carbon steel and anti-corrosion stainless steel, depending on the different application environments. Figure 6-45 shows the 2D drawings and the dimensional descriptions of the hex bolt.

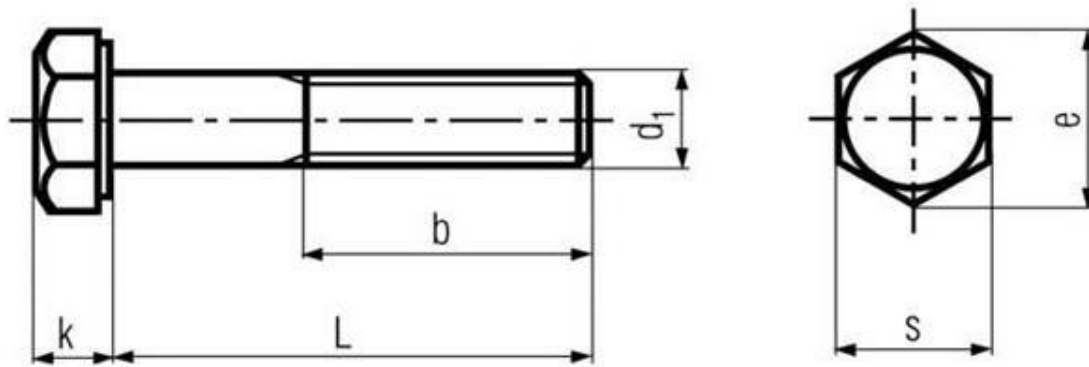


Figure 6-45: 2D drawings of the hex bolt with dimensional descriptions

As the hex bolt is a standardised part, existing product information can be collected from the existing hex bolt standard. In this use case, the hex bolt is selected from DIN (Deutsches Institut für Normung or German Institute for Standardisation) 931 standard in the literature. Table 6-8 shows the dimensions of the hex bolt in the DIN 931 standard.

Table 6-8: Hex bolt dimensions (in millimetres) – DIN 931 (partial)

Thread size D1	Threaded shank length b (L* < 125)	Threaded shank length b (L - 125 to 200)	Threaded shank length b (L > 200)	Head depth k	Width across corner e	Width across flats s (preferred size)
M10	26	32	45	6.4	18.9	17
M12	30	36	49	7.5	21.1	19
M14	34	40	53	8.8	24.49	22
M16	38	44	57	10	26.75	24
M18	42	48	61	11.5	30.14	27
M20	46	52	65	12.5	33.53	30

M22	50	56	69	14	35.72	32
M24	54	60	73	15	39.98	36

*L: Bolt length

For this use case implementation, an M12 hex bolt is selected with a bolt length of 80 mm. According to DIN 931, the thread length b is 30 mm. If the D1 of this hex bolt is being changed, all the other dimensional parameters have to be modified based on the standard correspondingly. Two rules that constrained dimensional parameters are extracted from the DIN 931 standard and defined as follows:

- Bolt rules 1 – when D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.
- Bolt rules 2 – when D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.

The existing information of an M12 hex bolt in this use case is shown in Table 6-9.

Table 6-9: Existing information of use case 2 – hex bolt

VPM knowledge class	Existing product information
Product	Hex bolt
Feature	hexagonal head and external male thread
Description	Fastener - hex bolt DIN 931
Function	Fasten
Behaviour	The bolt head locks the bolt in the place and nut is applied at the end.
Form	Cylinder with a hexagonal head and external male thread
Material	Alloy steel
Design intent	Fasten
Geometry	From STEP file
Dimension	D1: M12, L=80 mm, b =30mm, k=7.5 mm, e =21.1 mm, s=19

	mm.
Rules	<p>Bolt rules 1 – when D1 is M12, k should be 7.5, e should be 21.1 and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.</p> <p>Bolt rules 2 – when D1 is changed to M14, k should be 8.8, e should be 24.49 and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.</p>
Fit	Fit with hole and nut.
Constraint	None
Relationship	None
Reference	DIN 931

6.4.2 Measurement Parameters and Testing Scenarios

Similar to the previous use case, measurement parameters are defined for evaluation with the hex bolt (as shown in Table 6-10).

Table 6-10: Measurement parameters and expected results of use case 2- hex bolt

Measurement parameter	Evaluation criteria	Explanation	Expected results
Generative representation	C1	If the VPM can develop a model as a generative representation of the hex bolt	VPM product model structure of the hex bolt
Knowledge capture	C2	If the knowledge capture tool can capture the existing hex bolt information as existing knowledge and generate a knowledge file	Knowledge file in XML
Product geometry and knowledge visualisation	C3	If the interface can visualise the part geometry (from the step file) and its associated knowledge (from the	Geometry visualisation of the hex bolt in the interface. Knowledge

		knowledge file)	visualisation of the knowledge file in the interface.
Product relationship representation	C4	There is no relationship between this hex bolt and other parts because it is a single part and does not belong to any assembly.	The hex bolt is shown as a “part” in the interface.
Knowledge reasoning and reuse	C5	If the hex bolt dimensions are driven and constrained by rules with knowledge reasoning.	Constrain the change of hex bolt dimension by rules with knowledge reasoning.
Correctness of the changes	C6	If changes applied to the hex bolt geometry through reuse of the existing knowledge are correct.	Propagate the change of dimension correctly.
Data exchange of geometry	C7	If the hex bolt geometric data is exchanged through the STEP file.	Geometry visualisation in the interface.
Data exchange of knowledge	C8	If the captured knowledge of the hex bolt can be exchanged through the hex bolt knowledge file.	Modify the knowledge file, re-import it and present the changes of knowledge in the knowledge file

The following testing scenarios are identified to verify and validate the tool’s effectiveness:

- Scenario One – changing bolt length (L), threaded shank length (b), head depth (k), width across corner (e), width across flats (s) while keeping the thread size (D1) consistent.
- Scenarios Two – changing thread size (D1).

6.4.3 Virtual Product Modelling Framework Application

After defining the measurement parameters and testing scenarios, the next step is to utilise the hex bolt data as the input to evaluate the virtual product modelling framework. The following sections will focus on each step of the proposed virtual product modelling framework in application to this hex bolt.

a) Product model development

As shown in Figure 6-46, the hex bolt can be represented using the VPM product model structure in the same way described in use case 1.

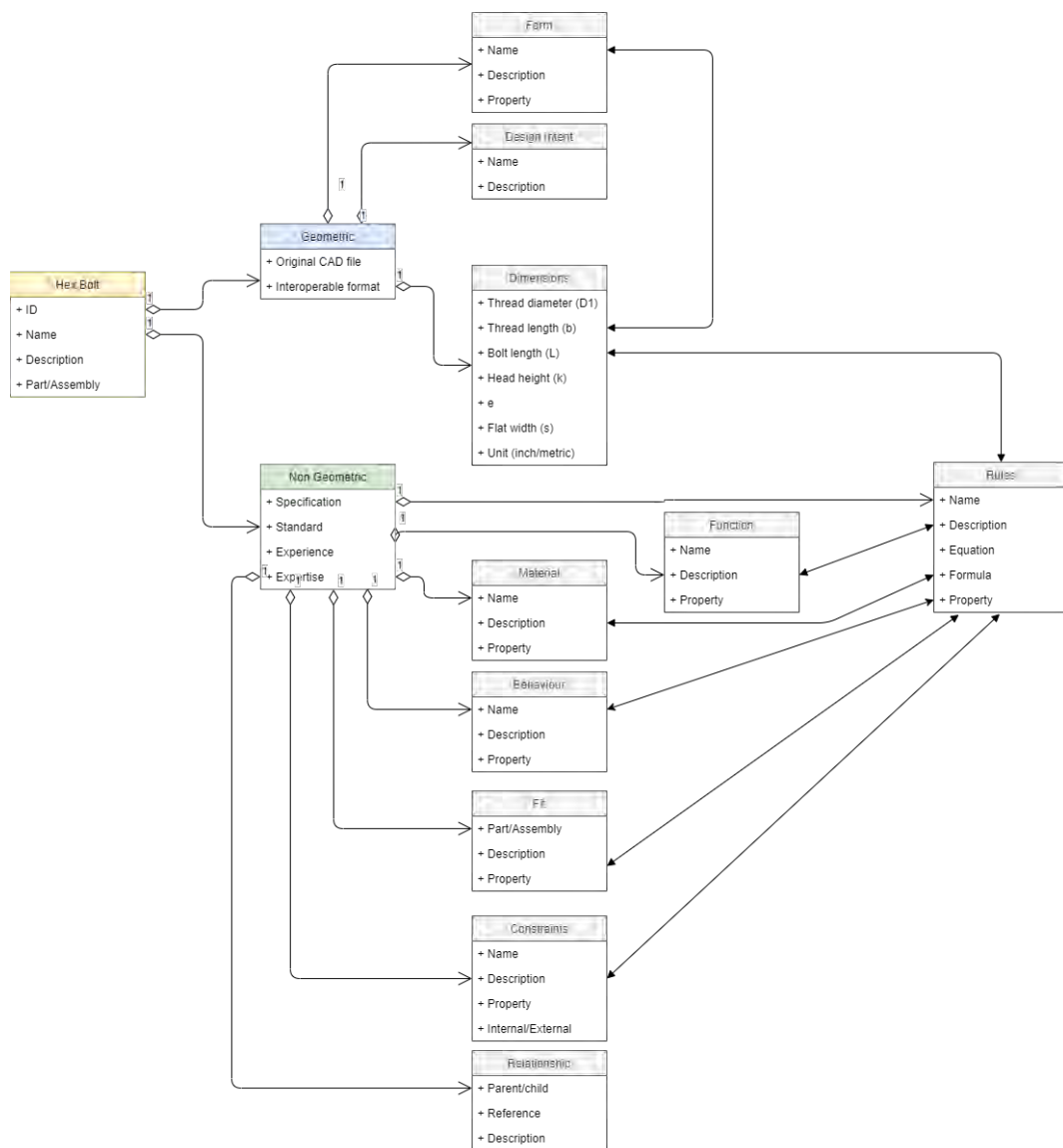


Figure 6-46: VPM product model structure of the hex bolt in UML diagram

b) Knowledge capture of non-geometric information

The existing non-geometric information of the hex bolt is captured in the same way as presented in Section 6.3.2.2 (b). The knowledge file of the hex bolt generated from the knowledge capture tool is shown in Figure 6-47.

```
<knowledge>
  <product type="part">
    <id>00000005</id>
    <name>Hex bolt</name>
    <description>Fastener - hex bolt DIN 931</description>

    <design_intent>
      <name>Fasten</name>
      <description>Fastener</description>
    </design_intent>

    <function_>
      <name>Fasten</name>
      <description>None</description>
      <property>None</property>
    </function_>

    <form>
      <name>Cylinder, hexagonal head, thread</name>
      <description>Cylinder with a hexagonal head and external male thread</description>
    </form>

    <material>
      <name>alloy steel</name>
      <description>The hex bolt is made of alloy steel</description>
      <property>None</property>
    </material>

    <behaviour>
      <name>Fasten</name>
      <description>The bolt head locks the bolt in the place and nut is applied at the end.</description>
      <property>None</property>
    </behaviour>

    <fit>
      <name>thread fit</name>
      <description>Fit with hole and nut.</description>
      <property>None</property>
    </fit>

    <relationship>
      <parent>None</parent>
      <children>None</children>
      <reference>None</reference>
      <description>None</description>
    </relationship>

    <rules>
      <name>rule01</name>
      <description>When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm.
        If L is less than 125 mm, the thread length b should be 30 mm;
        if L is between 125 mm and 200 mm, the thread length b should be 36 mm;
        if L is larger than 200 mm, the thread length b should be 49mm.
      </description>
      <formula></formula>
      <equation></equation>
    </rules>

    <rules>
      <name>rule02</name>
      <description>When D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm.
        If L is less than 125 mm, the thread length b should be 34 mm;
        if L is between 125 mm and 200 mm, the thread length b should be 40 mm;
        if L is larger than 200 mm, the thread length b should be 53 mm.
      </description>
      <formula></formula>
      <equation></equation>
    </rules>
  </product>
</knowledge>
```

Figure 6-47: Example - part of knowledge file of the hex bolt

The hex bolt can be further represented with detailed knowledge (shown in Figure 6-48).

d) Knowledge mapping

For this hex bolt part, two rules are captured and stored in the knowledge file, which are:

- Bolt rule 1 – when D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.
- Bolt rule 2 – when D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.

The knowledge mapping is performed by converting these rules into object-oriented programming logic that constrains the hex bolt dimension parameters. Later, the programmed logic will be used as the backend of the user interface. The mapping process between the knowledge file and the user interface for visualisation for “Scenario one – changing L under one consistent D1” is shown in Figure 6-49. In this use case, the changing of parameter L is mapped with the bolt rule 01. When the user changes the L value, the algorithm will check the input L value with the L value from the pre-defined rule 01. Three conditions are set up in the bolt rules. Based on bolt rule 01, if the input bolt length L value is less than 125, the thread length b has to be changed to 30. Similarly, if the input L value is in the range between 125 and 200, the thread length b has to be changed to 36. If the input L value is larger than 200, the thread length b has to be changed to 49. Therefore, when the user is changing the L parameter, the interface will indicate that the thread length b needs to be changed as well.

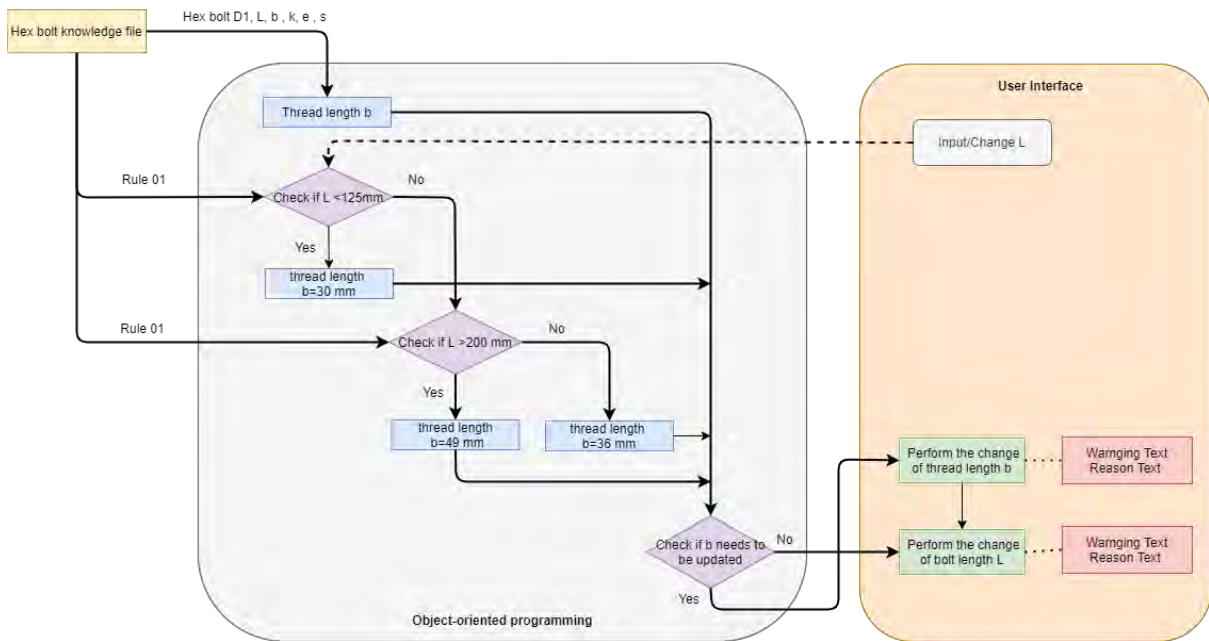


Figure 6-49: Illustration of the knowledge mapping for the scenario one – changing L under one D1 in use case 2.

The knowledge mapping process for scenario one – changing b under on D1 is shown in Figure 6-50. From the bolt rule 01, it can be known that the b value is constrained by the bolt length L and should be either 30, 36 or 49 for an M12 DIN 931 hex bolt. When the users change the b value in the user interface, the algorithm will check the b value against the bolt rule 01. If the input b value is among 30, 36 and 49, the user interface will tell the users that the bolt length L has to be changed accordingly. If the input b value is not in the given range from the bolt rule 01, the user interface will show that the value is conflicting with the rule through a warning text.

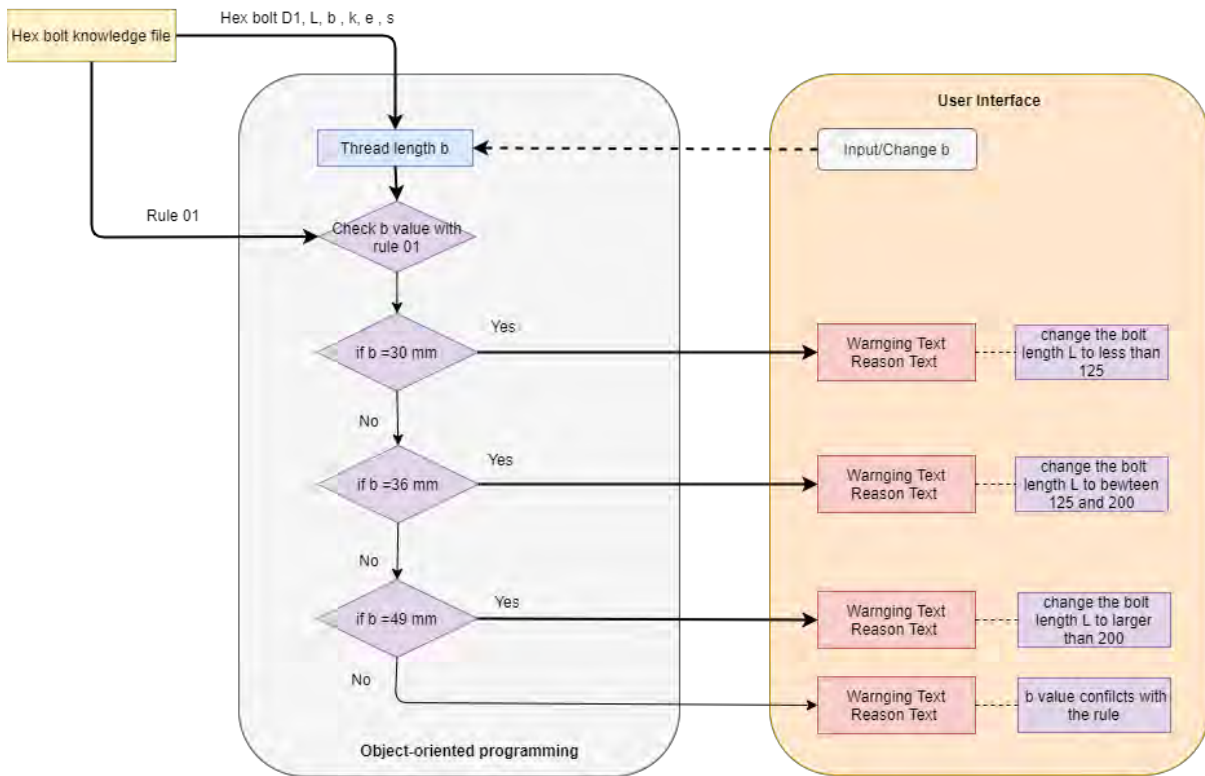


Figure 6-50: Illustration of the knowledge mapping for the scenario one – changing b under one D1 in use case 2.

The knowledge mapping process of scenario one- changing k, e, s under one D1 in use case 2 is shown in Figure 6-51. This mapping process is straightforward since the k, e and s values are all constants for an M12 hex bolt. If different values are given by the users, the algorithm will indicate that values are not allowed because the k, e and s are constrained by the bolt rule 01.

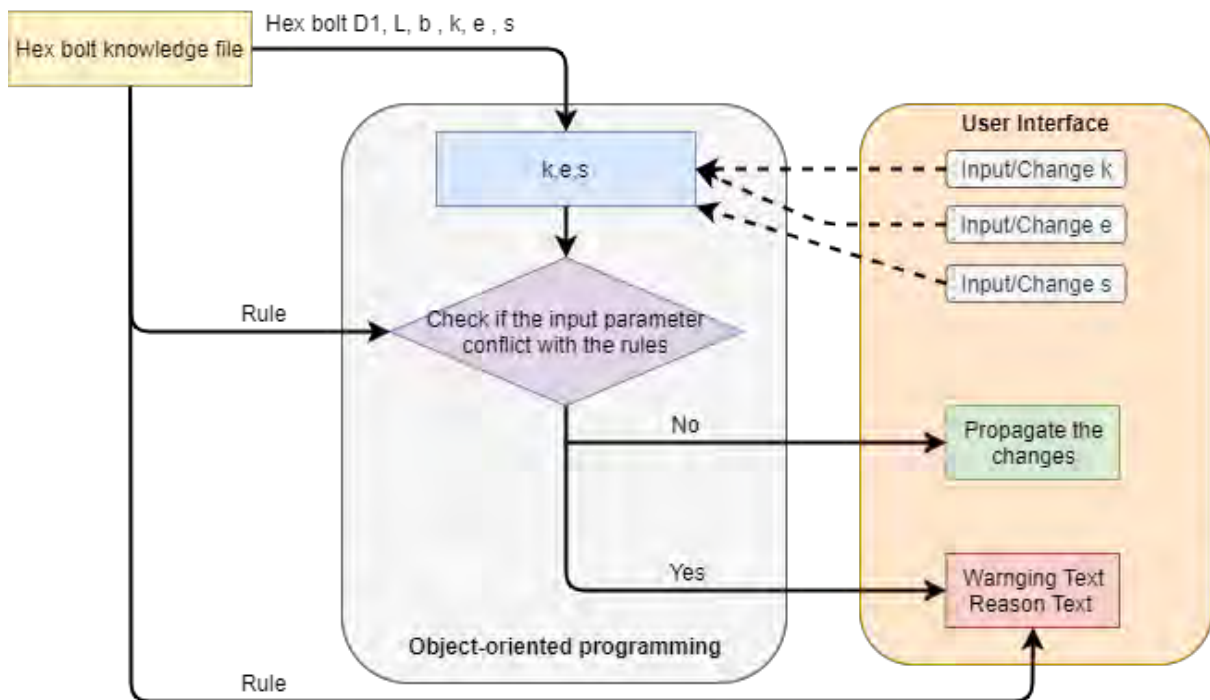


Figure 6-51: Illustration of the knowledge mapping for the scenario one – changing b under one D1 in use case 2.

The knowledge mapping process for scenario two is illustrated in Figure 6-52. When the user inputs D1 values, the algorithm will check the k, e, s values from the embed rules. If the users input k, e and s values, the algorithm will compare the input values with the values specified in the rules and then suggest if the changes can be made. Similarly, the algorithm will check the input L and b values with the rules, tell users whether these values are allowed by the rules, and show the relevant reasons why the changes cannot be performed. The full codification is provided in Appendix.

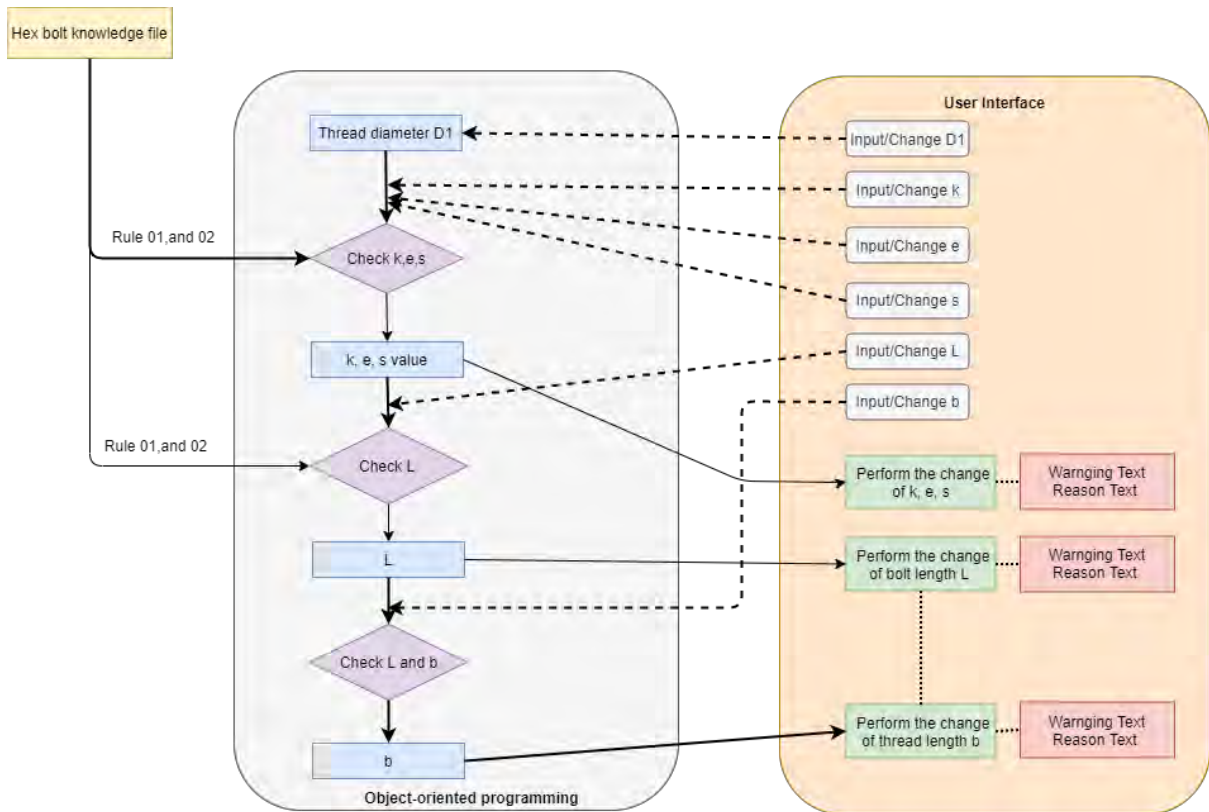


Figure 6-52: Illustration of the knowledge mapping for the scenario two – changing D1 in use case 2.

The knowledge mapping process provides the capability of knowledge reasoning for the users when they modify the hex bolt's dimension. After the knowledge mapping between the backend knowledge file and the frontend user interface, the next step is to visualise and validate the implementation results of use case 2.

e) Visualisation & validation

The 3D model of this hex bolt is visualised after importing the STEP file of the part into this tool. This visualisation process is completed the same way as described in Section 6.3.2.2 (e). The result of visualisation is shown in the following Figure 6-53. Two testing scenarios for this use case have been defined in Section 6.4.2 as follows:

- Scenario One – changing L, b, k e s under one consistent D1 (bolt rule 1).
- Scenarios Two – changing D1 (bolt rule 1 and rule 2).

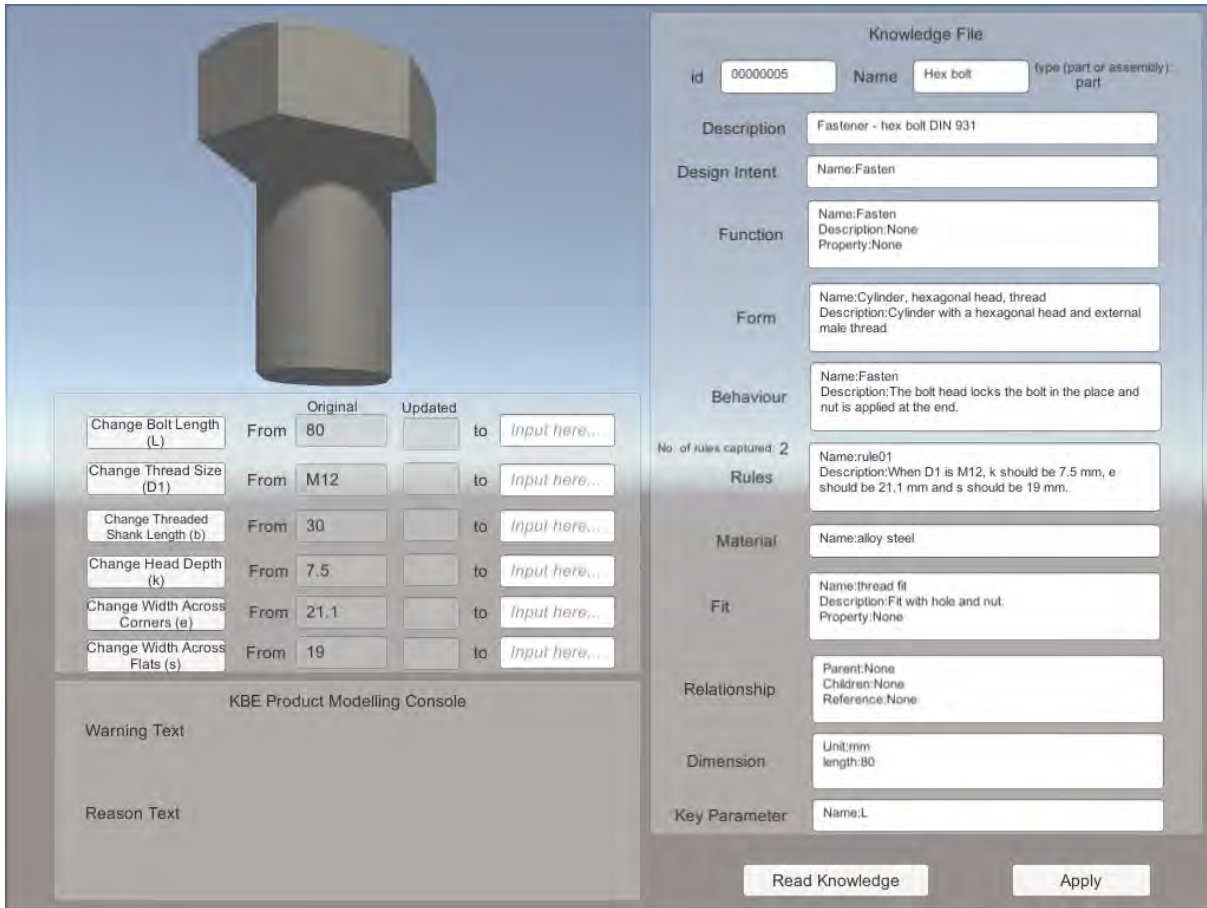


Figure 6-53: Hex bolt model visualised in the developed knowledge-based product modelling environment

The validation is performed along with the process of visualisation. For instance, after the design engineer selects to change the bolt length by clicking the button “Change Bolt Length” in the tool interface, the tool will indicate if the changes can be made and show the relevant changes that need to be considered. The results of changing bolt length under three conditions from bolt rule 01 are shown in Figure 6-54.

	Original	Updated	
Change Bolt Length (L)	From 80	70	to 70
Change Thread Size (D1)	From M12		to Input here...
Change Threaded Shank Length (b)	From 30	30	to Input here...
Change Head Depth (k)	From 7.5		to Input here...
Change Width Across Corners (e)	From 21.1		to Input here...
Change Width Across Flats (s)	From 19		to Input here...

KBE Product Modelling Console

You can make this change. You have changed L to 70 .You have to change the b to 30

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(a) $L < 125$ mm

	Original	Updated	
Change Bolt Length (L)	From 80	130	to 130
Change Thread Size (D1)	From M12		to Input here...
Change Threaded Shank Length (b)	From 30	36	to Input here...
Change Head Depth (k)	From 7.5		to Input here...
Change Width Across Corners (e)	From 21.1		to Input here...
Change Width Across Flats (s)	From 19		to Input here...

KBE Product Modelling Console

You can make this change. You have changed L to 130 .You have to change the b to 36

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(b) L between 125 and 200 mm

	Original	Updated	
Change Bolt Length (L)	From 80	240	to 240
Change Thread Size (D1)	From M12		to Input here...
Change Threaded Shank Length (b)	From 30	49	to Input here...
Change Head Depth (k)	From 7.5		to Input here...
Change Width Across Corners (e)	From 21.1		to Input here...
Change Width Across Flats (s)	From 19		to Input here...

KBE Product Modelling Console

You can make this change. You have changed L to 240 .You have to change the b to 49

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(c) $L > 200$ mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-54: Results of validation – use case 2: hex bolt, scenario one - change L

The validation of changing bolt threaded shank length b in scenario one is shown in Figure 6-55. In this case, four conditions were tested, and the results are as same as what has been claimed by bolt rule 01.

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="50"/>
Change Head Depth (k)	From 7.5		to <input type="text" value="input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="input here..."/>

KBE Product Modelling Console

You cannot make this change. b value conflict with rule01

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(a) b conflicts with rule

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="input here..."/>
Change Threaded Shank Length (b)	From 30	30	to <input type="text" value="30"/>
Change Head Depth (k)	From 7.5		to <input type="text" value="input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="input here..."/>

KBE Product Modelling Console

You can make this change. You have to change L to less than 125 mm

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(b) b =30 mm

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="input here..."/>
Change Threaded Shank Length (b)	From 30	36	to <input type="text" value="36"/>
Change Head Depth (k)	From 7.5		to <input type="text" value="input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="input here..."/>

KBE Product Modelling Console

You can make this change. You have to change L to between 125 and 200 mm

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm, if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(c) b =36 mm

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="input here..."/>
Change Threaded Shank Length (b)	From 30	49	to <input type="text" value="49"/>
Change Head Depth (k)	From 7.5		to <input type="text" value="input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="input here..."/>

KBE Product Modelling Console

You can make this change. You have to change L to larger than 200 mm

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm, if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(d) b =49 mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-55: Results of validation – use case 2: hex bolt, scenario one - change b

The validation results of changing bolt head depth k in scenario one are presented in Figure 6-56. If the input k value differs from the value allowed by rule01, the tool will tell the users that the change is not permitted.

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="8"/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="Input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="Input here..."/>

KBE Product Modelling Console

You cannot make this change. k value conflict with rule01

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm, if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(a) k conflicts with rule

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5	<input type="text" value="7.5"/>	to <input type="text" value="7.5"/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="Input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="Input here..."/>

KBE Product Modelling Console

You can make this change.

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm, if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(b) k =7.5mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-56: Results of validation – use case 2: hex bolt, scenario one - change k

Similarly, the validation of changing bolt width across corners (e) and changing bolt width across flats (s) have been done, and the results are shown in Figure 6-57 and Figure 6-58, respectively.

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="Input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="25"/>
Change Width Across Flats (s)	From 19		to <input type="text" value="Input here..."/>

KBE Product Modelling Console

You cannot make this change. e value conflict with rule01

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(a) e conflicts with rule

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="Input here..."/>
Change Width Across Corners (e)	From 21.1	<input type="text" value="21.1"/>	to <input type="text" value="21.1"/>
Change Width Across Flats (s)	From 19		to <input type="text" value="Input here..."/>

KBE Product Modelling Console

You can make this change.

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(b) e=21.1 mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-57: Results of validation – use case 2: hex bolt, scenario one - change e

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="Input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="Input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="30"/>

KBE Product Modelling Console

You cannot make this change. s value conflict with rule01

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(a) s conflicts with rule

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="Input here..."/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="Input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="Input here..."/>
Change Width Across Flats (s)	From 19	<input type="text" value="19"/>	to <input type="text" value="19"/>

KBE Product Modelling Console

You can make this change.

When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.

(b) s=19 mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-58: Results of validation – use case 2: hex bolt, scenario one - change s

Scenario two was tested by changing thread size D1 from M12 to M14. As shown in Figure 6-59, only when the b, k, e and s are all changed to the values permitted by the rules will the change of D1 from M12 to M14 be allowed to be performed.

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12		to <input type="text" value="M14"/>
Change Threaded Shank Length (b)	From 30		to <input type="text" value="Input here..."/>
Change Head Depth (k)	From 7.5		to <input type="text" value="Input here..."/>
Change Width Across Corners (e)	From 21.1		to <input type="text" value="Input here..."/>
Change Width Across Flats (s)	From 19		to <input type="text" value="Input here..."/>

KBE Product Modelling Console

You cannot make this change.

When D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.

(a) b, k, e, s values conflict with rule

	Original	Updated	
Change Bolt Length (L)	From 80		to <input type="text" value="Input here..."/>
Change Thread Size (D1)	From M12	<input type="text" value="M14"/>	to <input type="text" value="M14"/>
Change Threaded Shank Length (b)	From 30	<input type="text" value="34"/>	to <input type="text" value="34"/>
Change Head Depth (k)	From 7.5	<input type="text" value="8.8"/>	to <input type="text" value="8.8"/>
Change Width Across Corners (e)	From 21.1	<input type="text" value="24.4"/>	to <input type="text" value="24.49"/>
Change Width Across Flats (s)	From 19	<input type="text" value="22"/>	to <input type="text" value="22"/>

KBE Product Modelling Console

You can make this change.

When D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.

(b) b, k, e, s values allowed by rule

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Blue box - knowledge reasoning.

Figure 6-59: Results of validation – use case 2: hex bolt, scenario two – change bolt thread size D1 from M12 to M14

6.4.4 Result Analysis and Use Case Discussion

Similar to use case 1, the next evaluation objective is to critically analyse the virtual product modelling framework results in use case 2 and then compare the product modelling results from using the virtual product modelling framework with the use of the current existing/legacy product modelling system for the same circumstances. The virtual product modelling framework implementation in the hex bolt example further proves that the framework satisfied all these measurement parameters identified before (shown in Table 6-10).

1) Generative representation – C1

A product model structure is developed using VPM to provide a generative representation of the hex bolt using the VPM knowledge classes (as shown in Figure 6-38). The developed VPM product model structure shows the capability to capture and store the hex bolt's existing information for knowledge reuse in the product modelling process. Different from the previous use case, this hex bolt has more existing product information due to its complexity and standardisation. Therefore, the VPM product model structure has shown a higher level of generalisation of the hex bolt as more parameters of the hex bolt provide more product design configurations. The integration of design rules from the hex bolt standard allows this product model to be generalised to more product variants with different thread sizes, bolt lengths, threaded shank lengths, etc. Using this VPM product model structure can help save the time spent on designing hex bolt variants as the hex bolt model can be varied by changing parameters for a quick generation of product variants that are accepted by the standard.

2) Knowledge capture – C2

The virtual product modelling methodology is successful in capturing all the existing product information of the hex bolt example. Compared with the first use case, additional information such as function, behaviour, and fit are captured because more data about the hex bolt is available from the existing knowledge source. Further, more complex rules that define different conditions for parameter changing of the hex bolt are also captured successfully into the knowledge file. This proves that the VPM is capable of capturing complex rules and data, given that they are provided in the existing design knowledge. The captured information such as function, behaviour and fit help the users to understand these corresponding aspects of the hex bolt. This knowledge capture process ensures that all the existing product information of this hex bolt is captured and stored in a knowledge file. As a result, the design knowledge can be integrated with the developed VPM product model structure, and the knowledge file can be used as an accessible knowledge base that provides the users with available hex bolt data. It provides well-defined knowledge classes and a formalised knowledge capture method for users to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats. This will help save the time of retrieving product data from different knowledge sources.

3) Product geometry and knowledge visualisation – C3

The hex bolt geometry and the captured knowledge are visualised successfully in the interface. This proves that the implementation of VPM on this hex bolt is effective in terms of geometry and knowledge visualisation. The developed knowledge-based product modelling environment using VPM shows the capability of visualising the geometry of basic engineering components (imported from the STEP file) and the associated knowledge (imported from the knowledge file).

The possible changes to the hex bolt dimensions that can be made by the users in this interface are limited to the parameters that are collected from the standard for this particular use case. Similar to use case 1, the results of changes in the hex bolt geometry are presented through the text description in the interface. As analysed in Section 6.3.6, the compromise in visualisation is due to the lack of available enabling technologies. However, this visualisation result is acceptable as the developed product modelling environment has shown VPM's effectiveness in propagating the changes of hex bolt by reusing the captured knowledge. Despite the limitation in graphically visualising the changes of hex bolt geometry, the achieved knowledge visualisation can provide users with additional information about the hex bolt, which would help them have a clearer understanding of the hex bolt besides viewing the geometry. It could help save the time that users spend on knowing the hex bolt and the relevant standard before starting the product modelling process.

4) Product relationship representation – C4

In this use case, the hex bolt is applied in VPM as a single part, and it does not have relationships with other parts. As shown from the visualisation results in Figure 6-44, this hex bolt is identified as a “part” in the developed knowledge-based product modelling environment. This result proves the effectiveness and correctness of the relationship representation of the hex bolt example.

5) Knowledge reasoning and reuse – C5

From the validation results of use case 2, it can be seen that the developed knowledge-based product modelling environment shows the capability of changing bolt length (L), threaded shank length (b), head depth (k), width across corner (e), width across flats (s) and thread size (D1) of the hex bolt using text description. Rules from the existing standard are reused in VPM for knowledge reasoning and are effective in driving and constraining the targeted parameters of the hex bolt. Compared with the simple rules applied in the first use case, the

involved rules in the second use case are more complex as they constrain parameters with different conditions. These complex rules show the internal connections and constraints between various parameters of this hex bolt. One parameter may be constrained by different parameters at the same time, and changing this parameter may require the changes of other parameters as well. For example, to change the bolt thread size D1, other parameters such as threaded shank length (b), head depth (h), width across corners (e) and width across flats (s) are required to be changed accordingly (as shown in Figure 6-50). Moreover, the change of the threaded shank length under one thread size is further constrained by the bolt length. The propagation of the affected changes of the hex model is displayed in the tool interface, and all the changes are provided with reasons. These reasons are based on the previously captured rules from the existing product information. The knowledge reasoning and reuse allow these internally constrained parameters of the hex bolt to be changed in one modelling process. It can help users to check the changes according to the standard and eliminate errors that would happen during the process of changing parameters that have different internal constraints from other parameters.

6) Correctness of the changes – C6

The affected changes are reflected by text description due to the lack of enabling technology. However, by comparing the resulted values in the text with the expected results from the pre-defined rules, it can be seen that the changes applied to the geometry of the hex bolt through the reuse of the existing knowledge are correct. This proves that the modelling of the hex bolt through the implementation of VPM is effective. It ensures the correctness of the product modelling by using the developed knowledge-based product modelling environment.

7) Data exchange – C7 and C8

The STEP file of the hex bolt model is imported into the developed tool and graphically visualised in the tool interface. Similarly, the knowledge file is loaded by the tool, and the

associated knowledge is displayed in the tool interface correctly. Therefore, the data exchange of the hex bolt geometry through the STEP file and the exchange of the hex bolt knowledge through the knowledge file are proved to be successful in this use case. This will guarantee that the hex bolt model developed from VPM can be used and transferred among different knowledge-based product modelling environments.

The visualisation and validation results from use case 2 further prove that the proposed virtual product modelling methodology satisfies all the measurement parameters. Additionally, a comparison of product modelling results of a hex bolt using the current existing/legacy CAD system and using VPM for the same circumstances is provided in Table 6-11.

Table 6-11: Comparison of the use case 2 implementation results between the existing/legacy CAD system and VPM

Evaluation Criteria	Existing/legacy CAD system implementation	Virtual Product Modelling framework implementation
The capability of generative representation of engineering products in VPM	Use template hex bolt model as a generative 3D model (geometry representation only and limited to proprietary format). Limited product information is provided.	Develop a VPM product model structure as a generative representation of the hex bolt (standardised format - UML). Can provide information, such as function, behaviour, design intent, material, fit, design rules to understand possible product design configurations.
The capability of capturing the product geometry and its associated knowledge from the existing product	Capture the hex bolt geometry through importing/exporting the model into standardised	Capture the hex bolt geometry through importing the model from the CAD systems in a standardised

information	format. Unable to capture the existing hex bolt knowledge during the product modelling process.	format. Capture existing hex bolt knowledge during the product modelling process through the knowledge capture tool.
The capability of visualising the product geometry and its associated knowledge	Visualisation of hex bolt geometry only	Visualisation of hex bolt original geometry and its associated knowledge. Text visualisation of the changes of the hex bolt geometry parameters.
The capability of presenting every part of the product and the relationships among them	The hex bolt model is shown as single part in the modelling hierarchy tree and there is no relationship with other parts.	The hex bolt model is shown as “part” in the proof-of-concept tool interface and there is no relationship with other parts.
The capability of propagating changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge	Manual tracking of changes of parameters. The change of hex bolt geometry is reflected in 3D visualisation graphically. No knowledge reasoning is provided when changing parameters in the CAD systems. Rules are embedded in template model but not available for reusing.	Automatic tracking of changes of parameters through the reuse of rules. Text visualisation of the affected parameters in the proof-of-concept tool interface. The change of hex bolt geometry is reflected through text description due to the limited available enabling tools. Knowledge reasoning is provided in the proof-of-concept tool interface by

		reuse of rules from the existing knowledge
The correctness of the changes applied to the product geometry by reuse of the existing knowledge	Manual check-up of the correctness of changes is required.	The changes applied to the hex bolt geometry is following the rules from the standard. The correctness is proved during the validation stage.
The capability and correctness of the product geometry data exchange between different platforms	The hex bolt geometric data is exchanged in the format of a STEP file correctly.	The hex bolt geometric data is exchanged in the format of a STEP file. The correctness is proved during the visualisation process.
The capability and correctness of the knowledge exchange through knowledge file	The existing/legacy CAD systems are not able to exchange knowledge between each other using a generalised format.	The knowledge is exchanged in a knowledge file in the format of XML. XML is a platform-independent neutral format for data communication. Knowledge exchange is proved during the implementation and validation stage.

The above critical analysis further proves the effectiveness of the framework in the chosen basic engineering part use case. When the product becomes more complex, the difference in product modelling between using VPM and existing/legacy CAD systems becomes more apparent. In this use case 2, a hex bolt is selected as a basic engineering part to assess the effectiveness of VPM. With richer data provided, the developed product model using VPM has shown a higher level of generalisation of the hex bolt. Compared with the simple rules utilised in the first use case, the applied bolt rules are more complex as they imply the

internal connections and constraints between different parameters of the hex bolt. Therefore, through knowledge reasoning and reuse in VPM, the design rules from the hex bolt standard are integrated into the developed hex model. And with richer product information captured, this VPM hex bolt model can be generalised to more product variants with different combinations of thread size, bolt length and threaded shank length, etc. Furthermore, knowledge reasoning and reuse in VPM allow these internally constrained hex bolt parameters to be changed in one modelling process. The affected change of the hex model is propagated and visualised in the tool interface, and all the changes are provided with reasons. This would help users check the correctness of changes according to the standard and eliminate errors during the modelling process. Additionally, the captured knowledge, including the rules, can be exchanged using a knowledge file generated by the knowledge capture tool. This ensures steady data communication between different knowledge-based product modelling environments. Moreover, the developed VPM hex bolt model with the knowledge file can serve as an accessible knowledge base that provides the users with available hex bolt data. It will help save the time of retrieving hex bolt data from different knowledge sources. In real life design scenario, a hex bolt used in one product may vary from the one used in the other product. With sufficient data captured, VPM can be used to develop one unified hex bolt model that can be generalised to apply to different products. Users would only need to choose different standards and set values according to their design specifications. It would save the time of creating hex bolt variants for their applications in different products.

These added values from using VPM show how modelling a hex bolt in VPM is different from using the existing/legacy CAD systems and how VPM could enhance the hex bolt modelling process to support design engineering automation. In the next section, a wheel

assembly will be used in case 3 to verify and validate the effectiveness of the virtual product modelling framework implementation in engineering assembly.

6.5 Use Case 3: Engineer Assembly – Wheel Assembly

6.5.1 Use Case Overview

The third use case selects a wheel assembly as an engineering assembly to validate the developed framework. A wheel assembly is a crucial part of most automotive and is typically composed of a tyre and a wheel. In the literature, wheel assembly has been widely used as a demonstrative model to explain the model structure, component relationships and complex parameter configurations (Object Management Group, 2015). The implementation of a wheel assembly model in this research aims to validate that the proposed virtual product modelling framework can also be applied to the engineering part with assembly relationships and internal and external parameter constraints. Figure 6-60 shows a wheel assembly modelled in Siemens NX software.

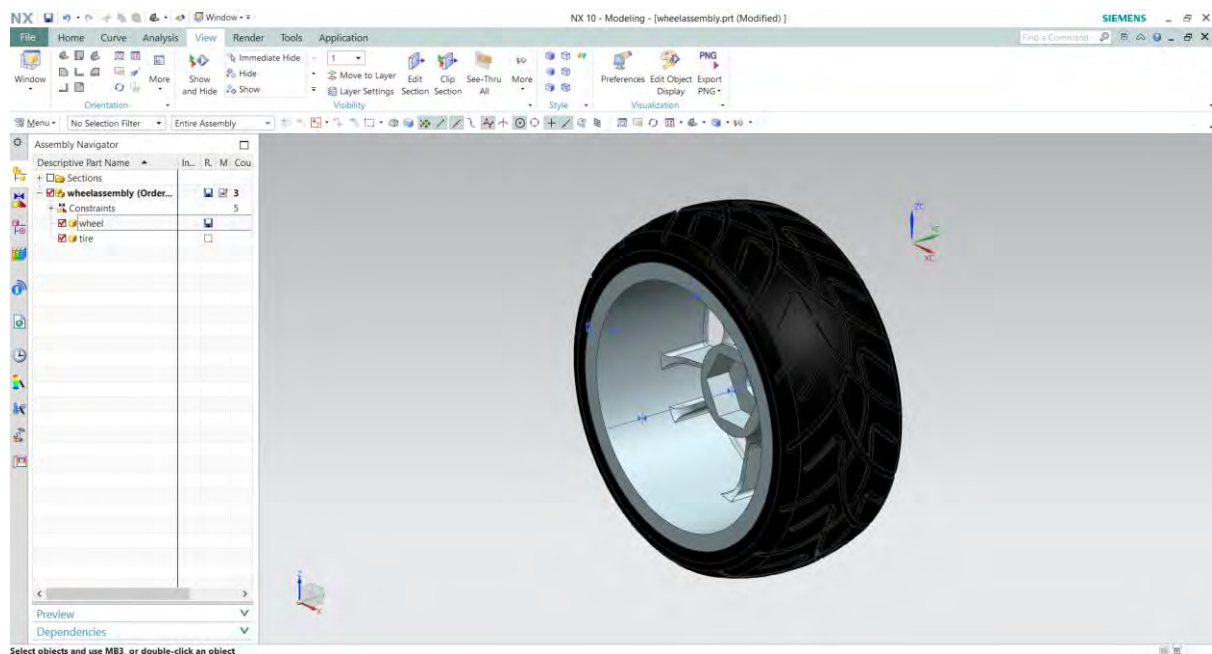


Figure 6-60: A wheel assembly modelled in Siemens NX 10

In this use case, the wheel assembly consists of two basic parts, which are a tyre and a wheel. In real life, when the tyre is inflated, the air pressure within the tyre keeps the tyre bead in the groove of the wheel. Theoretically, when the wheel assembly is designed, the dimensions of the tyre need to fit with the sizes of the wheel to ensure that these two parts are correctly assembled. Thus, in this use case, the existing product information can be extracted from the above theoretical assembling relations and dimensional parameters of the tyre and wheel parts. The following figures 6-61 and 6-62 show the dimensional parameters of the wheel and tyre parts, respectively.

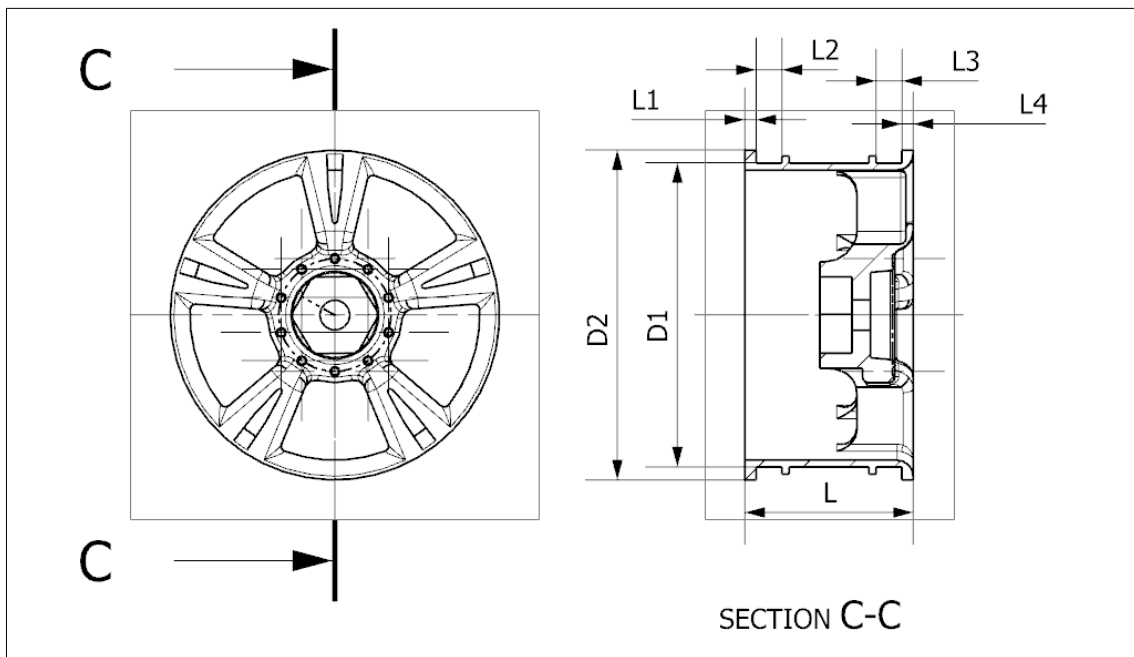


Figure 6-61: 2D drawings of the wheel part with dimensional descriptions

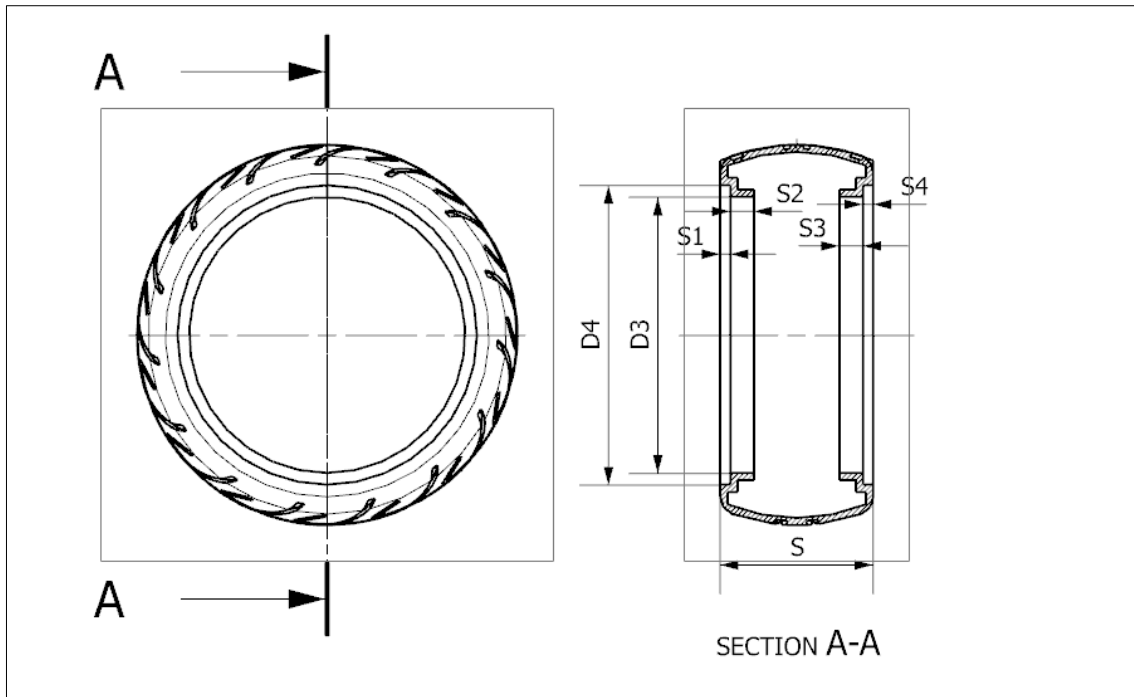


Figure 6-62: 2D drawings of the tyre part with dimensional descriptions

As mentioned previously in this section, when the wheel part and tyre part are assembled, all these assembling dimensions need to fit with each other. In this way, the rules of the wheel assembly are derived. The collected dimensional parameters and assembly rules, and part rules of the wheel assembly are shown in Table 6-12.

Table 6-12: Wheel assembly parameters and rules

Wheel Assembly	Wheel Assembly Parameter	Assembly Rules	Part Rules
Wheel	D1	$D1=D3$	-
	D2	$D2=D4$	-
	L	$L=S$	-
	L1	$L1=S1$	$L1=L4$
	L2	$L2=S2$	$L2=L3$
	L3	$L3=S3$	$L3=L2$
	L4	$L4=S4$	$L4=L1$
Tyre	D3	$D3=D1$	-
	D4	$D4=D2$	-
	S	$S=L$	-
	S1	$S1=L1$	$S1=S4$
	S2	$S2=L2$	$S2=S3$

	S3	S3=L3	S3=S2
	S4	S4=L4	S4=S1

In this use case implementation, the collected assembly rules and part rules will be used later in testing scenarios to validate the effectiveness of this framework. Wheel assembly rules are defined as follows:

- Wheel assembly rules 1: $D1 = D3$, when D1 is changed, D3 needs to be changed as well, and vice versa.
- Wheel assembly rules 2: $D2 = D4$, when D2 is changed, D4 needs to be changed as well, and vice versa.
- Wheel assembly rules 3: $L1 = S1$, when L1 is changed, S1 needs to be changed as well, and vice versa.
- Wheel assembly rules 4: $L2 = S2$, when L2 is changed, S2 needs to be changed as well, and vice versa.
- Wheel assembly rules 5: $L3 = S3$, when L3 is changed, S3 needs to be changed as well, and vice versa.
- Wheel assembly rules 6: $L4 = S4$, when L4 is changed, S4 needs to be changed as well, and vice versa.
- Wheel assembly rules 7: $L = S$, when L is changed, S needs to be changed as well, and vice versa.

Meanwhile, rules that should be applied when modelling each single part are defined and shown below:

- Wheel rules 1: $L1 = L4$, when L1 is changed, L4 needs to be changed as well, and vice versa.

- Wheel rules 2: $L2 = L3$, when $L2$ is changed, $L3$ needs to be changed as well, and vice versa.
- Tyre rules 1: $S1 = S4$, when $S1$ is changed, $S4$ needs to be changed as well, and vice versa.
- Tyre rules 2: $S2 = S3$, when $S2$ is changed, $S3$ needs to be changed as well, and vice versa.

The existing product information of the wheel assembly is shown in Table 6-13.

Table 6-13: Existing information of use case 3 – wheel assembly

VPM knowledge class	Existing product information
Product	Wheel assembly
Feature	Circular component, assembly
Description	Assembly of a wheel part and a tyre part
Function	Mounting and rotating for movement
Behaviour	Keeps wheel attached to a hub and the car axle.
Form	Circular
Material	Mixed
Design intent	Mounting and rotating for movement
Geometry	From STEP file
Dimension	D1:1884 mm D2:2046 mm D3:1884 mm D4:2046 mm L:1045 mm S:1045 mm L1:70 mm L2:160 mm L3:70 mm L4:160 mm S1:70 mm S2:160 mm S3:70 mm S4:160 mm
Rules	<ul style="list-style-type: none"> • Wheel assembly rules 1: $D1 = D3$, when $D1$ is changed, $D3$ needs to be changed as well, and vice versa. • Wheel assembly rules 2: $D2 = D4$, when $D2$ is changed, $D4$ needs to be changed as well, and vice versa. • Wheel assembly rules 3: $L1 = S1$, when $L1$ is changed, $S1$ needs to be changed as well, and vice versa. • Wheel assembly rules 4: $L2 = S2$, when $L2$ is changed, $S2$ needs to be changed as well, and vice versa. • Wheel assembly rules 5: $L3 = S3$, when $L3$ is changed, $S3$ needs to be changed as well, and vice versa. • Wheel assembly rules 6: $L4 = S4$, when $L4$ is changed, $S4$ needs to be changed as well, and vice versa. • Wheel assembly rules 7: $L = S$, when L is changed, S

	needs to be changed as well, and vice versa.
Fit	Tyre fits with wheel
Constraint	Assembly constraints
Relationship	Parent of the wheel part and tyre part
Reference	None

As this wheel assembly consists of two parts, information about the individual wheel part and tyre part is also collected and listed in Table 6-14 and Table 6-15. The existing knowledge of wheel assembly provides general information about the wheel assembly itself, while the existing data of each part offers more detailed information about the child part.

Table 6-14: Existing information of use case 3 – wheel

VPM knowledge class	Existing product information
Product	Wheel assembly part - wheel
Feature	Circular component
Description	Circular component in the wheel assembly. In conjunction with axles to rotate.
Function	rotating for movement
Behaviour	In conjunction with axles to rotate; mount with tyre
Form	Circular
Material	Alloy steel
Design intent	Mounting tyre and rotating for movement
Geometry	From STEP file
Dimension	D1:1884 mm D2:2046 mm L:1045 mm L1:70 mm L2:160 mm L3:70 mm L4:160 mm
Rules	<ul style="list-style-type: none"> • Wheel rules 1: $L1 = L4$, when $L1$ is changed, $L4$ needs to be changed as well, and vice versa. • Wheel rules 2: $L2 = L3$, when $L2$ is changed, $L3$ needs to be changed as well, and vice versa.
Fit	Wheel fits with tyre
Constraint	Assembly constraints
Relationship	Children of the wheel assembly
Reference	None

Table 6-15: Existing information of use case 3 – tyre

VPM knowledge class	Existing product information
Product	Wheel assembly part - tyre
Feature	Circular component
Description	Part of a wheel assembly, covering of a wheel.
Function	Transfer load, support, provide traction and cushion.
Behaviour	Surrounds a wheel to transfer a vehicle's load from the axle through the wheel to the ground and to provide traction on the surface over which the wheel travels; Also provide a flexible cushion
Form	Circular
Material	Synthetic rubber, natural rubber and fabric and wire, etc;
Design intent	Transfer load, support, provide traction and cushion.
Geometry	From STEP file
Dimension	D3:1884 mm D4:2046 mm S:1045 mm S1:70 mm S2:160 mm S3:70 mm S4:160 mm
Rules	<ul style="list-style-type: none"> • Tyre rules 1: $S1 = S4$, when S1 is changed, S4 needs to be changed as well, and vice versa. • Tyre rules 2: $S2 = S3$, when S2 is changed, S3 needs to be changed as well, and vice versa.
Fit	Tyre fits with wheel
Constraint	Assembly constraints
Relationship	Children of the wheel assembly
Reference	None

6.5.2 Measurement Parameters and Testing Scenarios

Measurement parameters are defined and explained in the following Table 6-16 for the evaluation of VPM in application to assembly and parts that have both internal and external parameter constraints.

Table 6-16: Measurement parameters and expected results of use case 3 – wheel assembly

Measurement parameter	Evaluation criteria	Explanation	Expected results
Generative representation	C1	If the VPM can develop a model as a generative representation of the wheel assembly, wheel part and tyre part.	VPM product model structure of the wheel assembly, wheel part and tyre part
Knowledge capture	C2	If the knowledge capture tool can capture the existing wheel assembly, wheel part and tyre information as existing knowledge and generate knowledge files of them	Three knowledge files in XML (for assembly, wheel part and tyre part)
Product geometry and knowledge visualisation	C3	If the interface can visualise the wheel assembly and parts' geometry (from the step file) and their associated knowledge (from the knowledge file)	Geometry visualisation of the wheel assembly (including the wheel and tyre part) in the interface. Knowledge visualisation of these knowledge files in the interface.
Product relationship representation	C4	If the interface can show the assembly relationship of the wheel assembly.	Show the assembly relationship – wheel assembly is shown as “assembly” while the wheel part and tyre part are shown as “part” in the interface.
Knowledge reasoning and	C5	If the dimensions of the wheel assembly (including parts) are	Constrain the change of the wheel assembly

reuse		driven and constrained by rules with knowledge reasoning.	(including parts) dimension by rules with knowledge reasoning.
Correctness of the changes	C6	If changes applied to the wheel assembly geometry (including parts) through reuse of the existing knowledge are correct.	Propagate the change of dimension correctly.
Data exchange of geometry	C7	If the geometric data of the wheel assembly and parts is exchanged through the STEP file.	Geometry visualisation in the interface.
Data exchange of knowledge	C8	If the captured knowledge of the wheel assembly (including parts) can be exchanged through knowledge files.	Modify the knowledge file, re-import it and present the changes of knowledge in the knowledge file

The following testing scenarios are identified to verify and validate the tool's effectiveness:

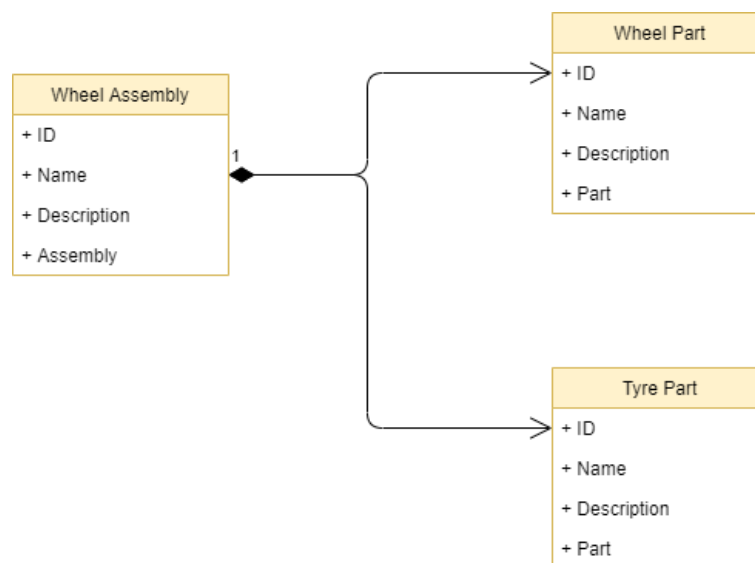
- Scenario One – changing wheel part dimension (with internal constraints from the wheel part itself and external constraints from the tyre part),
- Scenarios Two – changing the tyre part dimension (internal constraints from the tyre part itself and external constraints from the wheel).

6.5.3 Virtual Product Modelling Framework Application

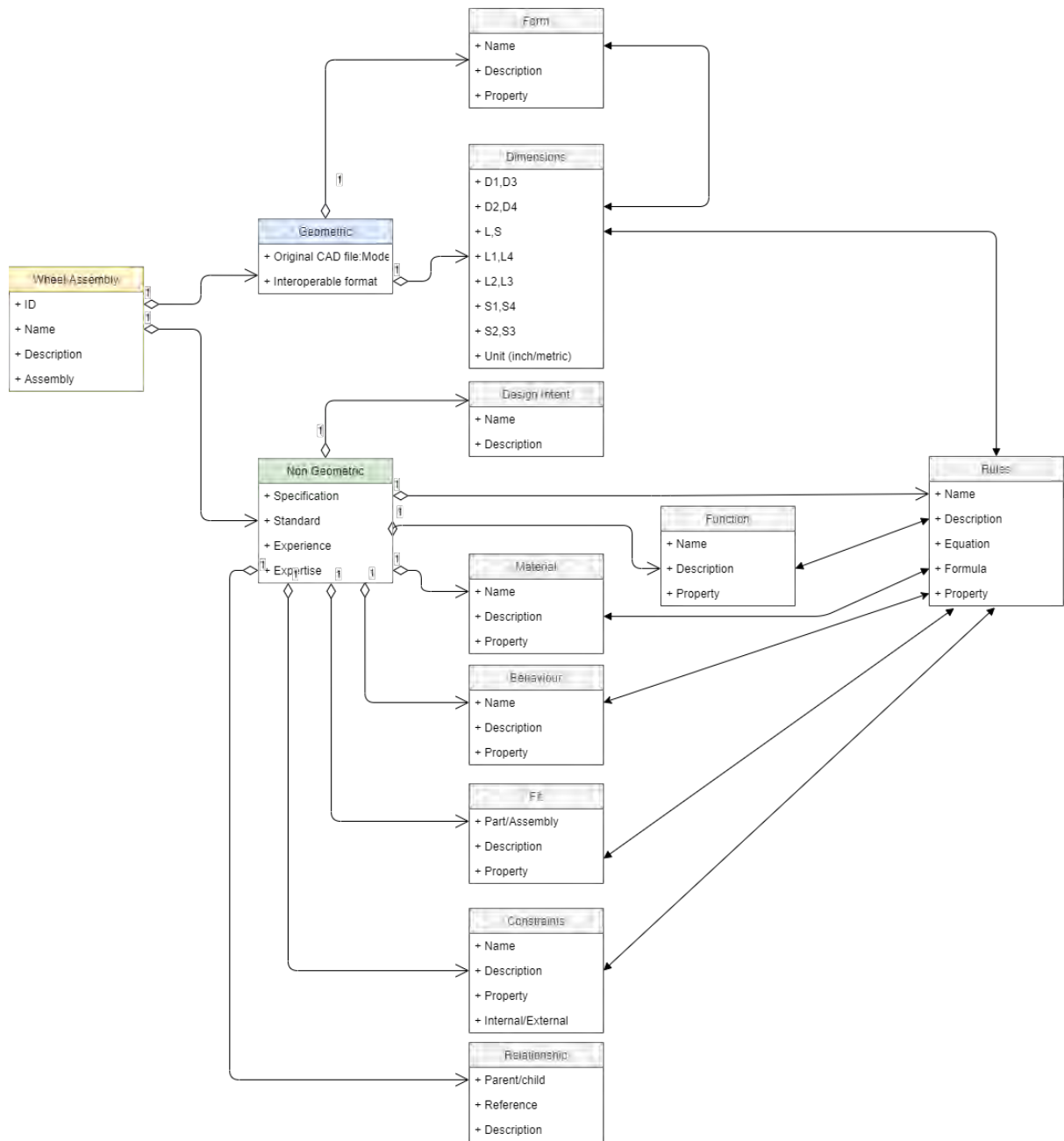
The overall VPM application steps in this use case are consistent with the steps described in the previous two use cases, and each step of VPM in application to this use case 3 is explained in the sections below.

a) Product model development

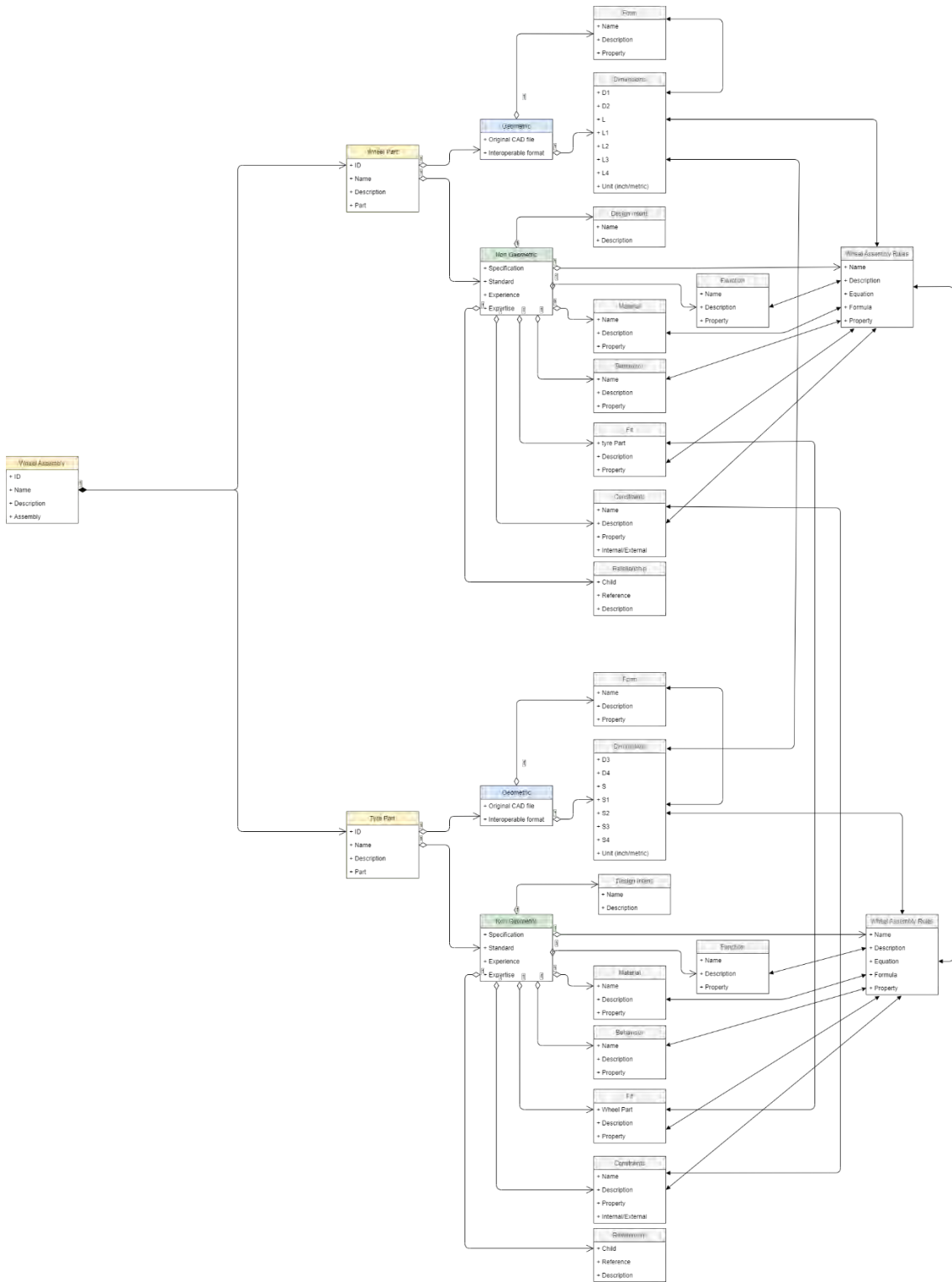
Similar to use cases 1 and 2, the wheel assembly can be represented with a VPM product model structure. In this use case, the wheel assembly consists of a wheel part and a tyre part. The UML diagram in Figure 6-63(a) shows the top level of the VPM product model structure of the wheel assembly model. It can be seen from Figure 6-63(a) that the wheel assembly has the assembly model itself as the parent and the wheel part and tyre part as children. The VPM product model structure of the assembly model itself is developed and shown in Figure 6-63(b). Moreover, a second level VPM product model structure that provides a detailed structure of child parts is developed and shown in Figure 6-63(c). Therefore, the top level of the wheel assembly VPM product model structure is expanded with the explicit atomic decomposition of product knowledge.



(a) Top level structure



(b) VPM product model structure of the wheel assembly itself



(c) Second level VPM product model structure of the wheel assembly product model

Figure 6-63: Virtual product model structure of the wheel assembly in UML diagram

b) Knowledge capture of non-geometric information

As explained in the previous section, the wheel assembly product model is represented by three VPM structures - the assembly model itself, the wheel part model and the tyre part model under the assembly model. The existing information of the wheel assembly model is input into the knowledge capture tool, and a knowledge file of the assembly in the format of XML is generated afterwards. Besides the wheel assembly model itself, knowledge files are also generated using the knowledge capture tool for the wheel and the tyre parts. The knowledge files of the assembly model, the wheel part model and the tyre part model are shown in Figure 6-64.

```
<knowledge>
  <product type="assembly">
    <id>00000000</id>
    <name>Wheel assembly</name>
    <description>Assembly of a wheel part and a tyre part</description>

    <design_intent>
      <name>Mounting tyre and rotating for movement</name>
      <description>Mounting tyre and rotating for movement</description>
    </design_intent>

    <function_>
      <name>Mounting and rotating for movement</name>
      <description>Mounting with axle and rotating for movement</description>
      <property></property>
    </function_>

    <form>
      <name>Circular</name>
      <description>Circular component</description>
    </form>

    <material>
      <name>Mixed</name>
      <description>Different material for wheel and tyre</description>
      <property></property>
    </material>

    <behaviour>
      <name>Mounting and rotating for movement</name>
      <description>Keeps wheel attached to a hub and the car axle.</description>
      <property></property>
    </behaviour>

    <fit>
      <name>Assembly fit</name>
      <description>Tyre fits with wheel</description>
      <property></property>
    </fit>

    <relationship>
      <parent>None</parent>
      <child>wheel part and tyre part</child>
      <reference></reference>
      <description>Parent of wheel part and tyre part</description>
    </relationship>
  </product>
</knowledge>
```

(a) Example - part of knowledge file of the wheel assembly

```

<knowledge>
  <product type="part">
    <id>00000006</id>
    <name>Wheel assembly part - wheel</name>
    <description>Circular component in the wheel assembly. In conjunction with axles to rotate.</description>

    <design_intent>
      <name>Mounting tyre and rotating for movement</name>
      <description>Mounting tyre and rotating for movement</description>
    </design_intent>

    <function_>
      <name>rotating for movement</name>
      <description>In conjunction with axle to rotate</description>
      <property>none</property>
    </function_>

    <form>
      <name>Circular</name>
      <description>Circular component</description>
    </form>

    <material>
      <name>Alloy steel</name>
      <description>none</description>
      <property>none</property>
    </material>
    <behaviour>
      <name>rotating for movement</name>
      <description>In conjunction with axles to rotate</description>
      <property></property>
    </behaviour>

    <fit>
      <name>Assembly fit</name>
      <description>Wheel fits with tyre</description>
      <property></property>
    </fit>

    <relationship>
      <parent>Wheel assembly</parent>
      <child>none</child>
      <reference></reference>
      <description>Child of the wheel assembly</description>
    </relationship>
  </product>
</knowledge>

```

(b) Example - part of knowledge file of the wheel assembly part – wheel

```

<knowledge>
  <product type="part">
    <id>00000007</id>
    <name>Wheel assembly part - tyre</name>
    <description>Part of a wheel assembly, covering of a wheel.</description>

    <design_intention>
      <name>Transfer load, support, provide traction and cushion.</name>
      <description>Transfer load, support, provide traction and cushion.</description>
    </design_intention>

    <function_>
      <name>Transfer load, support, provide traction and cushion.</name>
      <description>Transfer load, support, provide traction and cushion.</description>
      <property>none</property>
    </function_>

    <form>
      <name>Circular</name>
      <description>Circular component</description>
    </form>

    <material>
      <name>Synthetic rubber, natural rubber and fabric and wire, etc</name>
      <description>none</description>
      <property>none</property>
    </material>
    <behaviour>
      <name>Transfer load, support, provide traction and cushion.</name>
      <description>Surrounds a wheel to transfer a vehicle's load from the axle through the wheel to the ground,
        and to provide traction on the surface over which the wheel travels; Also provide a flexible cushion.
      </description>
      <property>none</property>
    </behaviour>

    <fit>
      <name>Assembly fit</name>
      <description>Tyre fits with wheel</description>
      <property>none</property>
    </fit>

    <relationship>
      <parent>Wheel assembly</parent>
      <child>none</child>
      <reference>none</reference>
      <description>Child of the wheel assembly</description>
    </relationship>
  </product>
</knowledge>

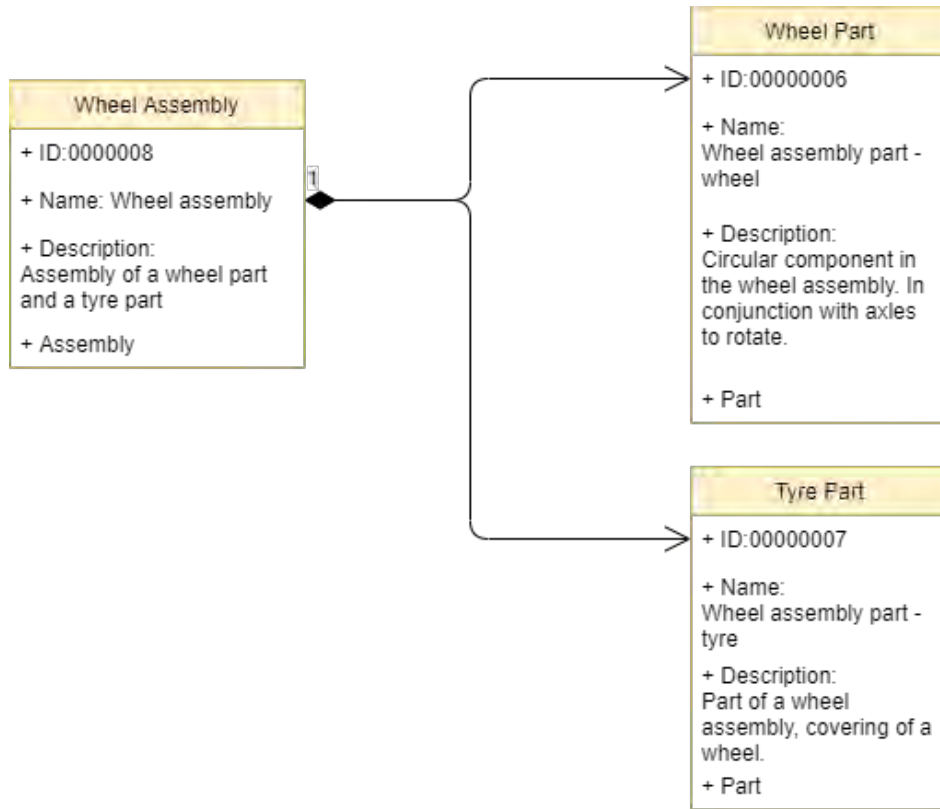
```

(c) Example - part of knowledge file of the wheel assembly part – tyre

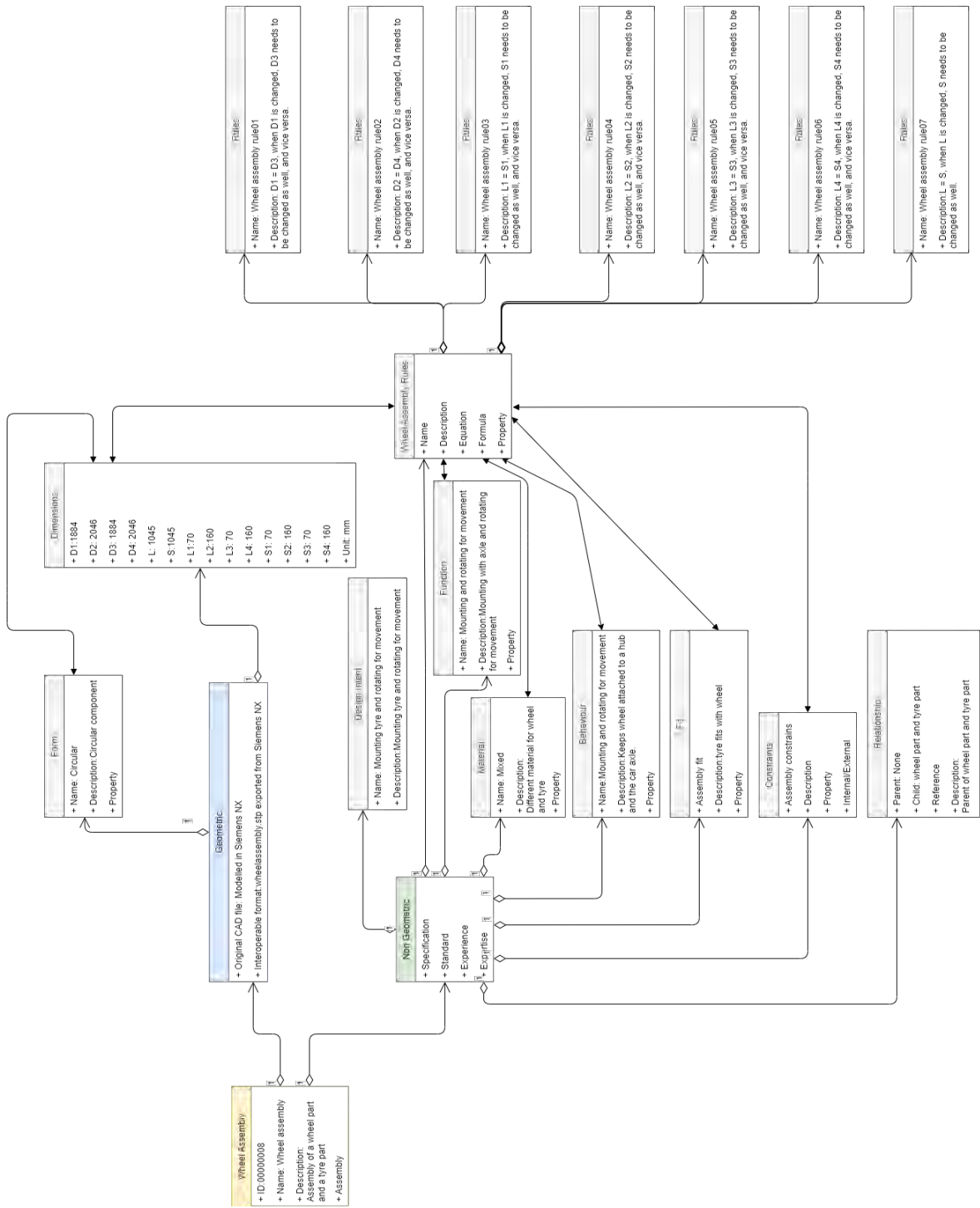
Figure 6-64: Examples - parts of knowledge files of the wheel assembly, wheel part and tyre

part

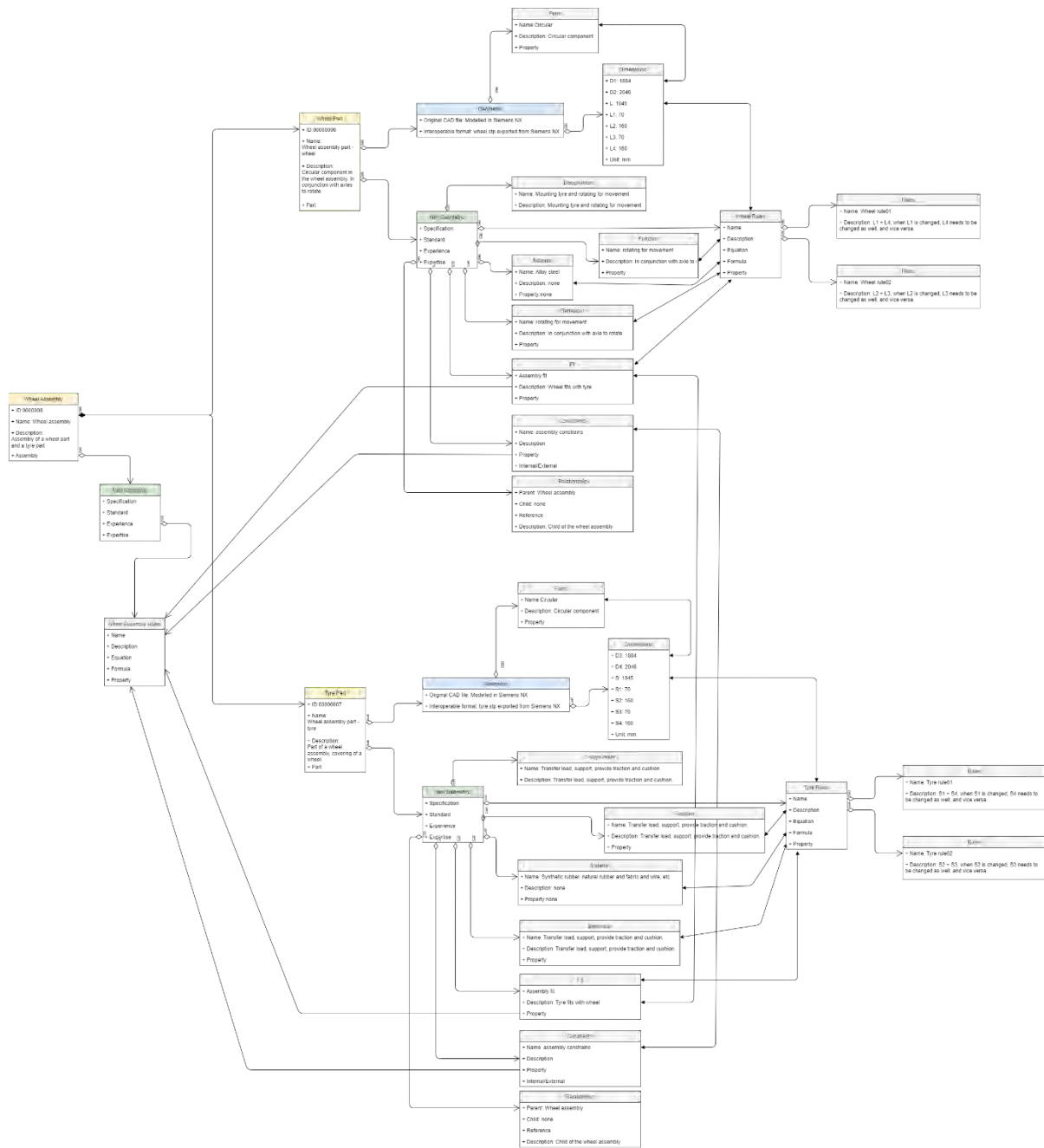
The VPM structure of the wheel assembly, wheel part and tyre part can be further developed by adding detailed knowledge into different atomic blocks (as shown in Figure 6-65).



(a) Top-level assembly structure with captured knowledge



(b) VPM product model structure of the wheel assembly itself with captured knowledge



(c) VPM product model structure of the wheel part and tyre part with captured knowledge

Figure 6-65: Virtual product models of the wheel assembly, wheel part and tyre part in UML diagram

c) Knowledge capture of geometry information

To visualise the wheel assembly product model represented by VPM, geometry information of the wheel assembly, wheel part and tyre part need to be extracted and stored into STEP files. This process is done through the following steps:

- Pre-model the wheel part in a CAD software Siemens NX 10 (Figure 6-66)
- Pre-model the tyre part in a CAD software Siemens NX 10 (Figure 6-67)
- Pre-model the wheel assembly by using the modelled wheel part and tyre part from the previous steps. (Figure 6-60)
- Export the wheel assembly model, wheel part model and tyre part model into STEP files using the export function in Siemens NX 10.

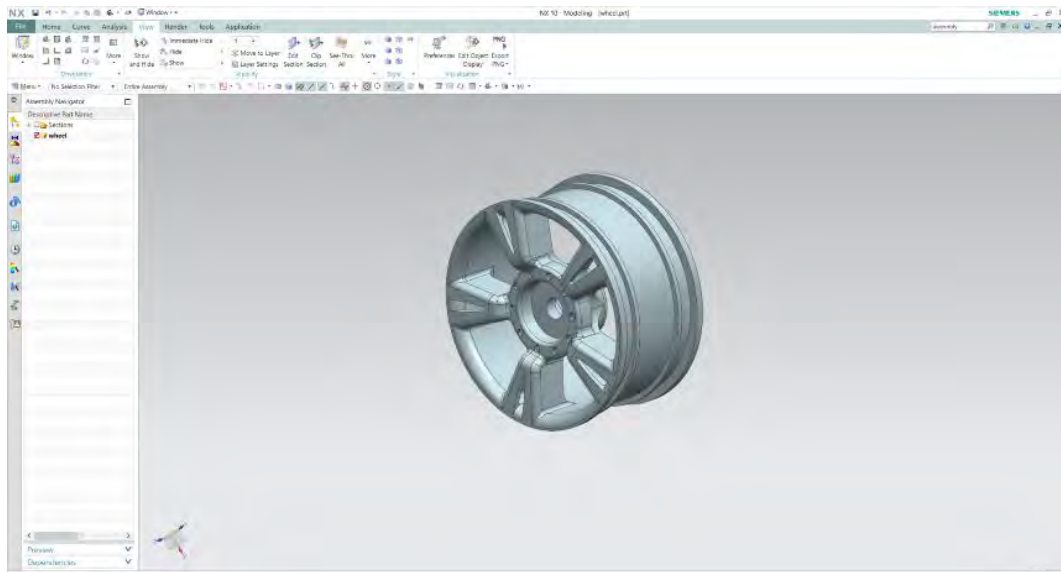


Figure 6-66: Wheel part modelled in Siemens NX 10

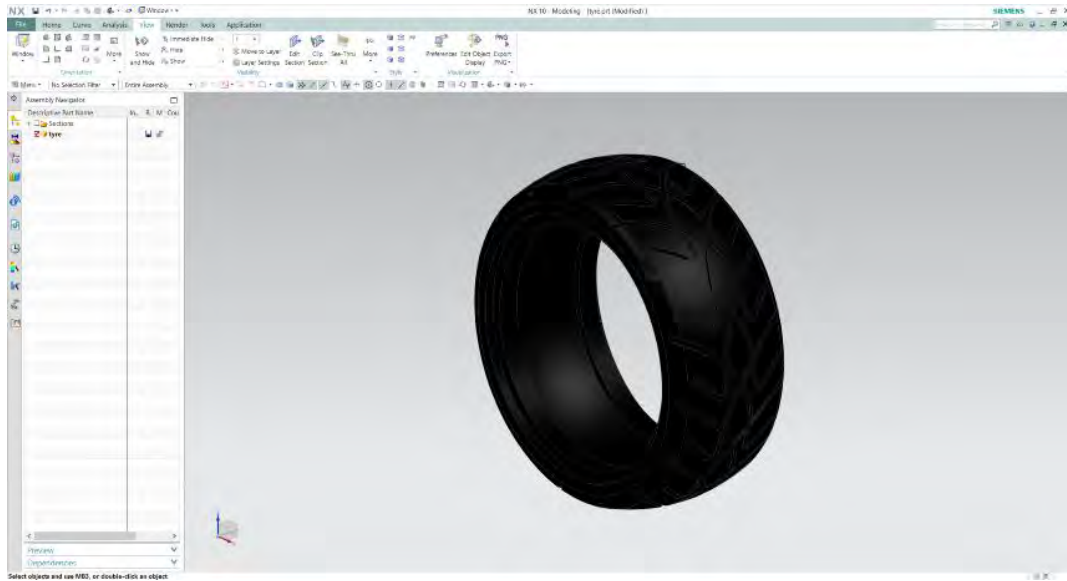


Figure 6-67: Tyre part modelled in Siemens NX 10

d) Knowledge mapping

As explained in previous sections, the knowledge mapping process is performed by converting rules into object-oriented programming logic that constrains the product model parameters. In this use case 3, the knowledge mapping stage includes two steps. The first step is to map the rules of individual parts in the assembly. For the wheel part, rules have been captured as follows:

- Wheel rules 1: $L1 = L4$, when $L1$ is changed, $L4$ needs to be changed as well, and vice versa.
- Wheel rules 2: $L2 = L3$, when $L2$ is changed, $L3$ needs to be changed as well, and vice versa.

Figure 6-68 shows the knowledge mapping process of the wheel part rules.

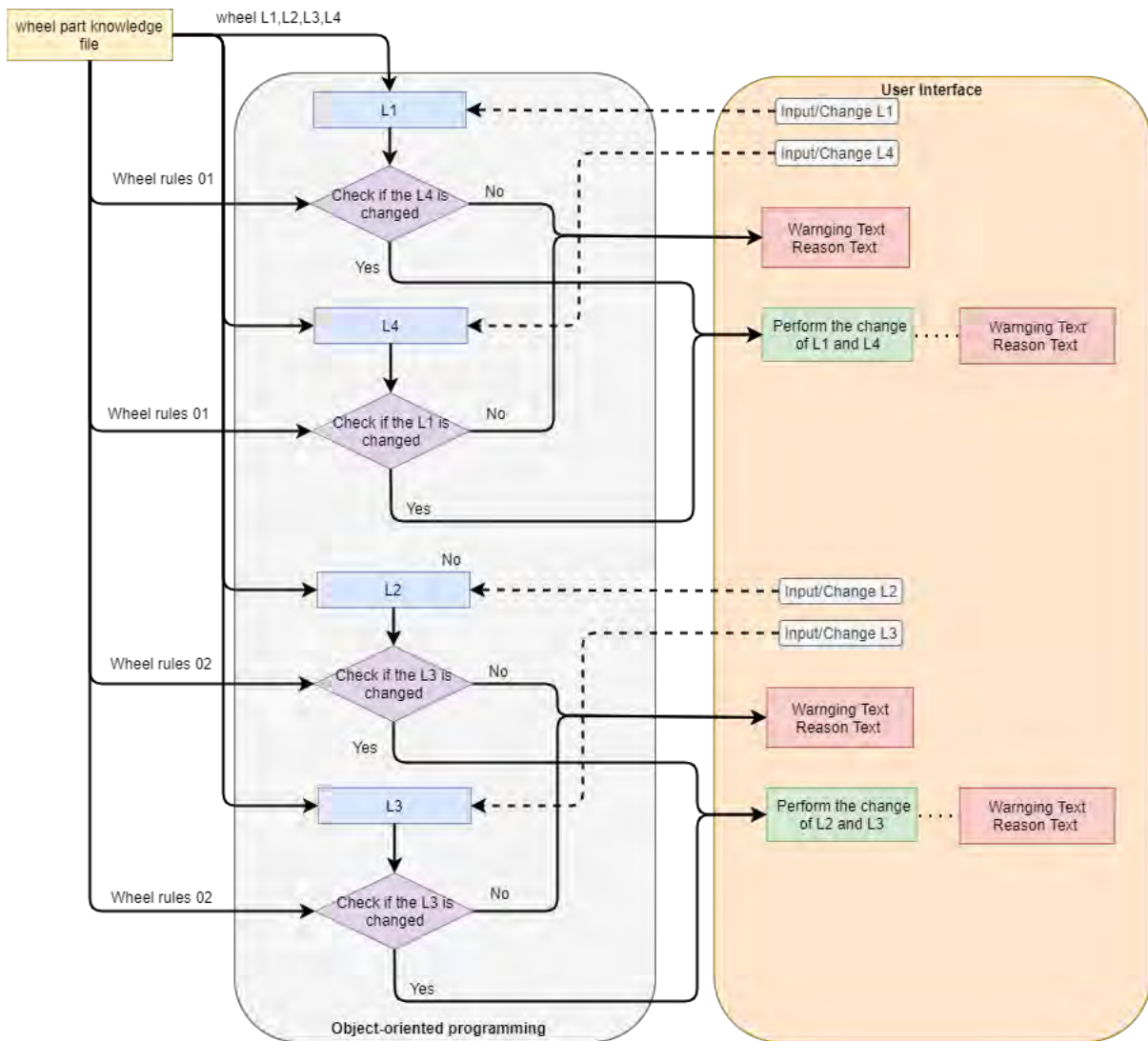


Figure 6-68: Illustration of the knowledge mapping for the wheel part rules

For the tyre part, rules have been captured as follows:

- Tyre rules 1: $S1 = S4$, when $S1$ is changed, $S4$ needs to be changed as well, and vice versa.
- Tyre rules 2: $S2 = S3$, when $S2$ is changed, $S3$ needs to be changed as well, and vice versa.

The knowledge mapping of tyre part rules (shown in Figure 6-69) is similar to the wheel part rule's mapping process.

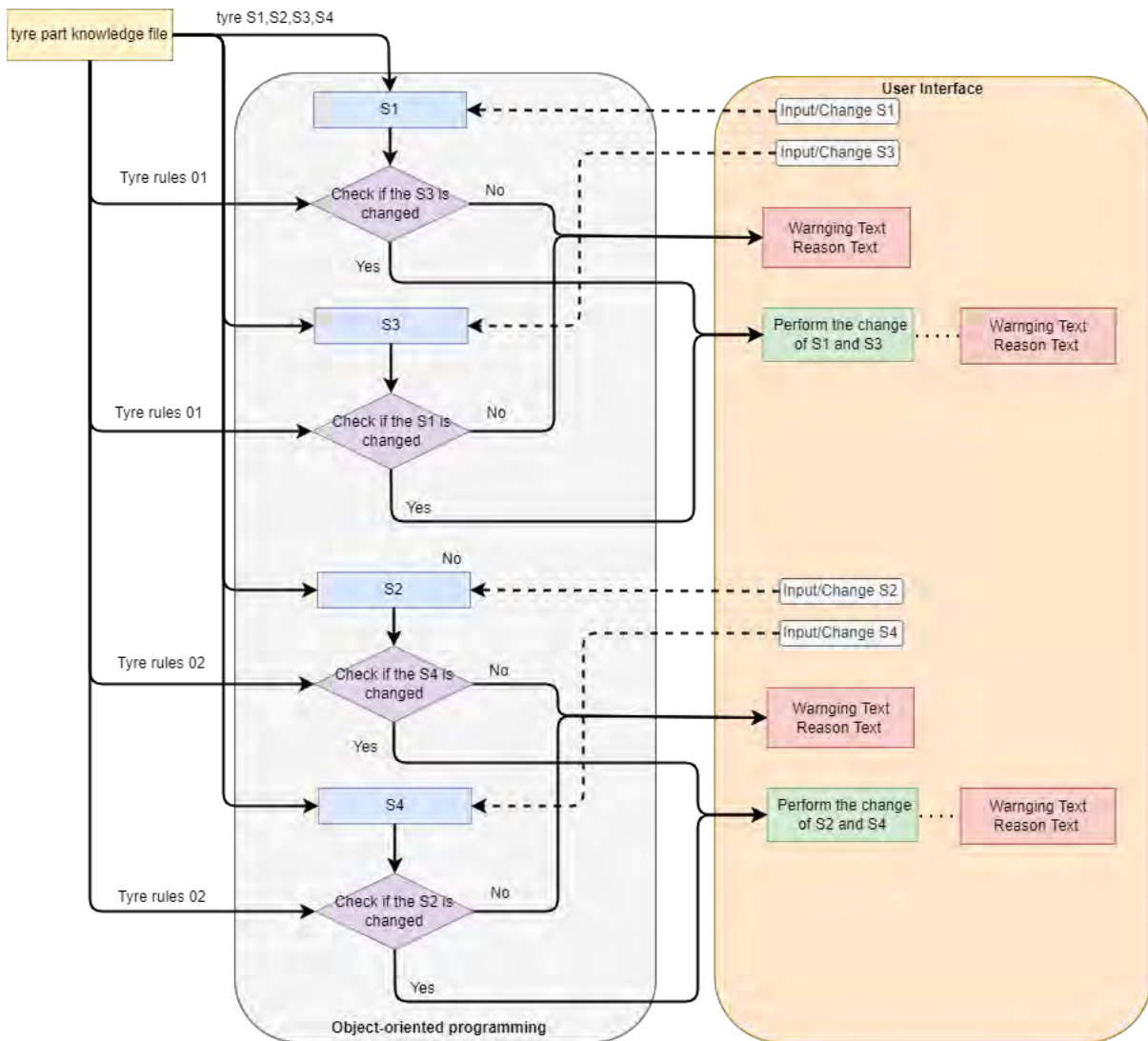


Figure 6-69: Illustration of the knowledge mapping for the tyre part rules

After mapping the individual part rules, the second step is to map the assembly rules of the wheel assembly. The captured assembly rules are listed below:

- Wheel assembly rules 1: $D1 = D3$, when $D1$ is changed, $D3$ needs to be changed as well, and vice versa.
- Wheel assembly rules 2: $D2 = D4$, when $D2$ is changed, $D4$ needs to be changed as well, and vice versa.
- Wheel assembly rules 3: $L1 = S1$, when $L1$ is changed, $S1$ needs to be changed as well, and vice versa.

- Wheel assembly rules 4: $L2 = S2$, when L2 is changed, S2 needs to be changed as well, and vice versa.
- Wheel assembly rules 5: $L3 = S3$, when L3 is changed, S3 needs to be changed as well, and vice versa.
- Wheel assembly rules 6: $L4 = S4$, when L4 is changed, S4 needs to be changed as well, and vice versa.
- Wheel assembly rules 7: $L = S$, when L is changed, S needs to be changed as well, and vice versa.

The knowledge mapping of wheel assembly rules is performed and explained in the following Figure 6-70.

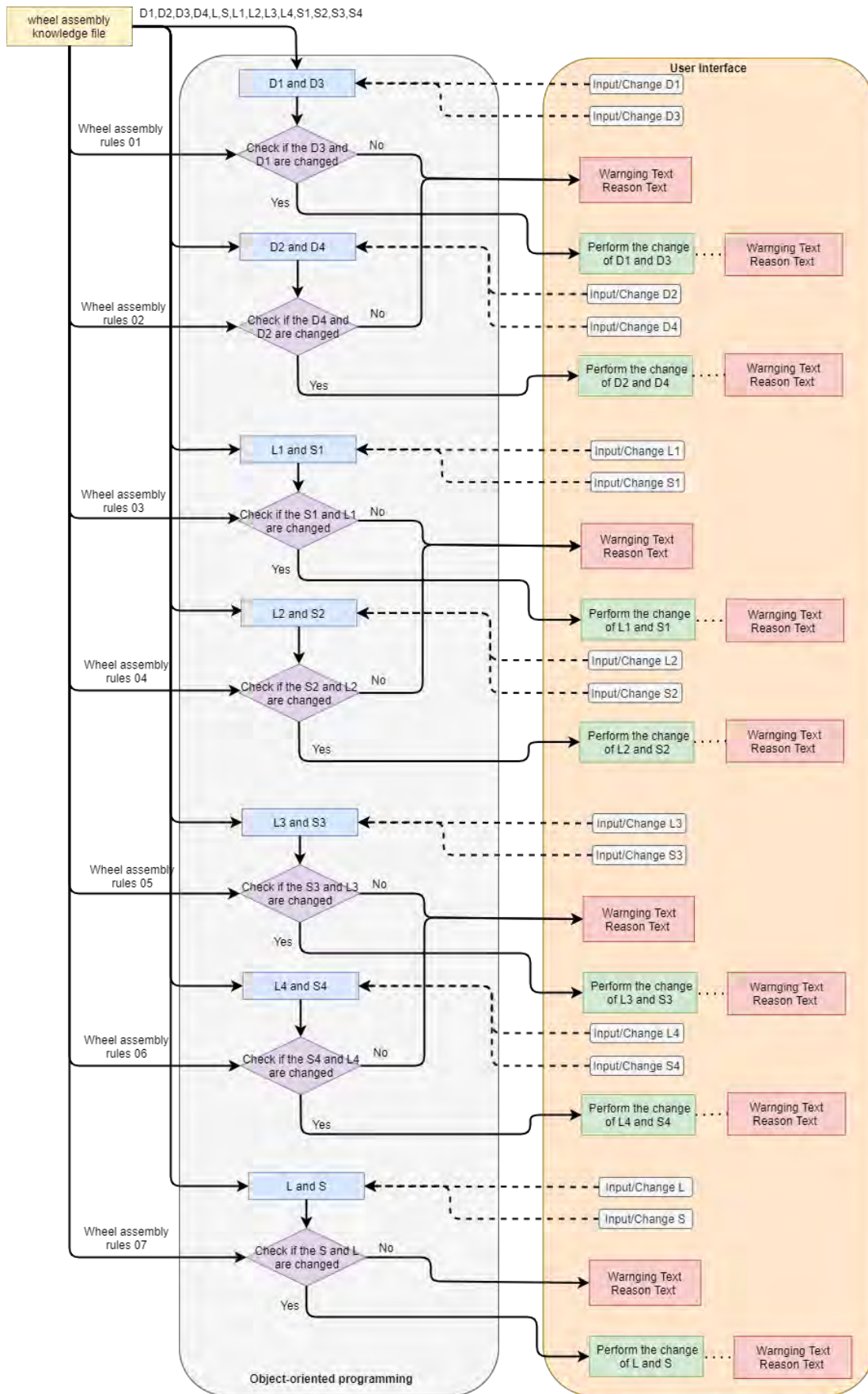


Figure 6-70: Illustration of the knowledge mapping for the wheel assembly rules

In testing scenarios one and two, the knowledge mapping process can be regarded as a combination of the knowledge mapping of wheel assembly rules, wheel part rules and part rules. This process is illustrated in the following Figure 6-71. The codification of knowledge mapping in use case 3 is provided in Appendix.

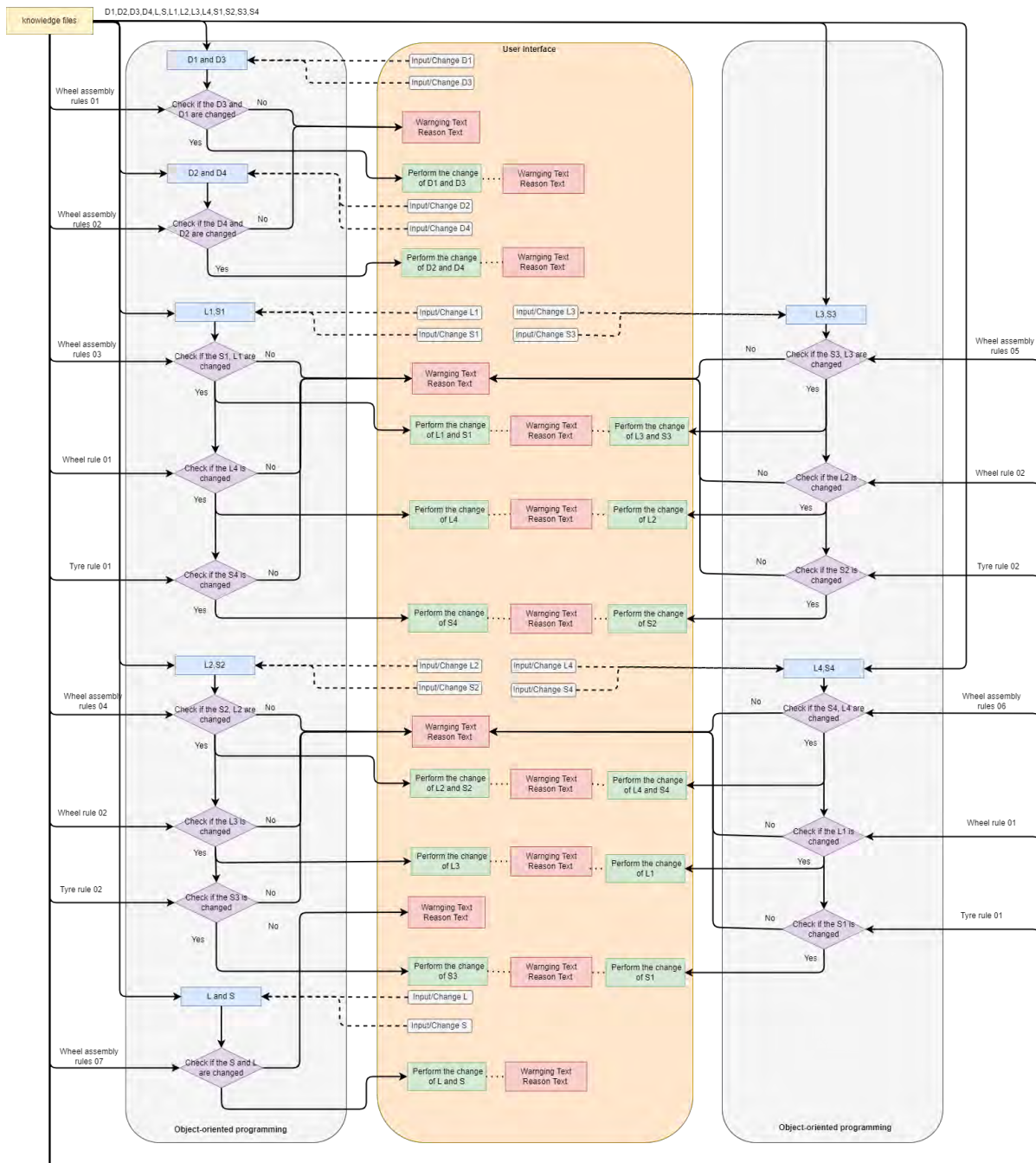


Figure 6-71: Illustration of the knowledge mapping for the testing scenario one and two in use case 3

e) Visualisation & validation

This visualisation process is completed the same way as described in Section 6.3.2.2 (e). After importing STEP files into the tool, the developed product modelling environment shows the ability to visualise the wheel assembly's original geometry, the wheel part's original geometry, and the tyre's original geometry. The knowledge of the wheel assembly stored in the knowledge file is also parsed and visualised in the tool interface (as shown in Figure 6-72). Further, functions of making possible changes to geometry are developed and shown in Figure 6-73. An example of the associated knowledge applied to constrain the change of wheel assembly geometry is also shown in Figure 6-73.

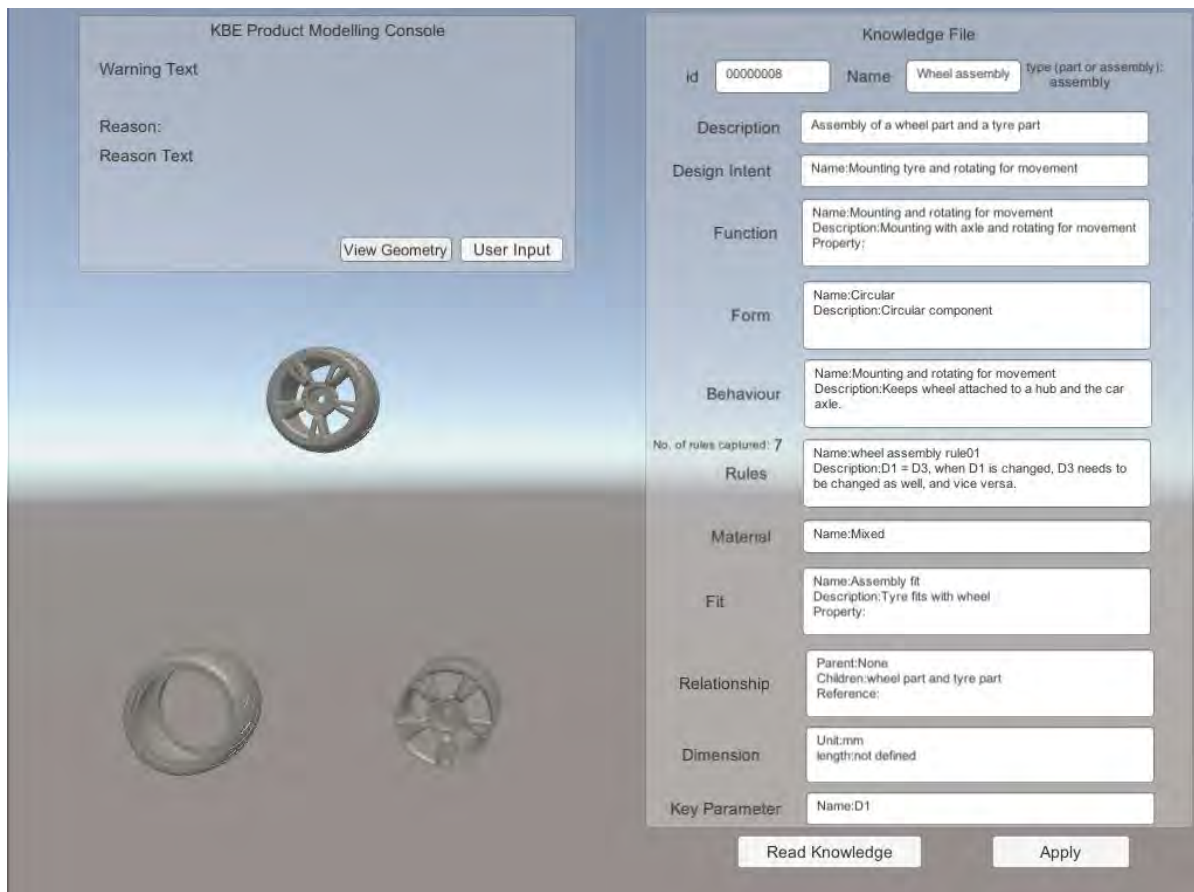


Figure 6-72: Wheel assembly model visualised in the developed knowledge-based product modelling environment

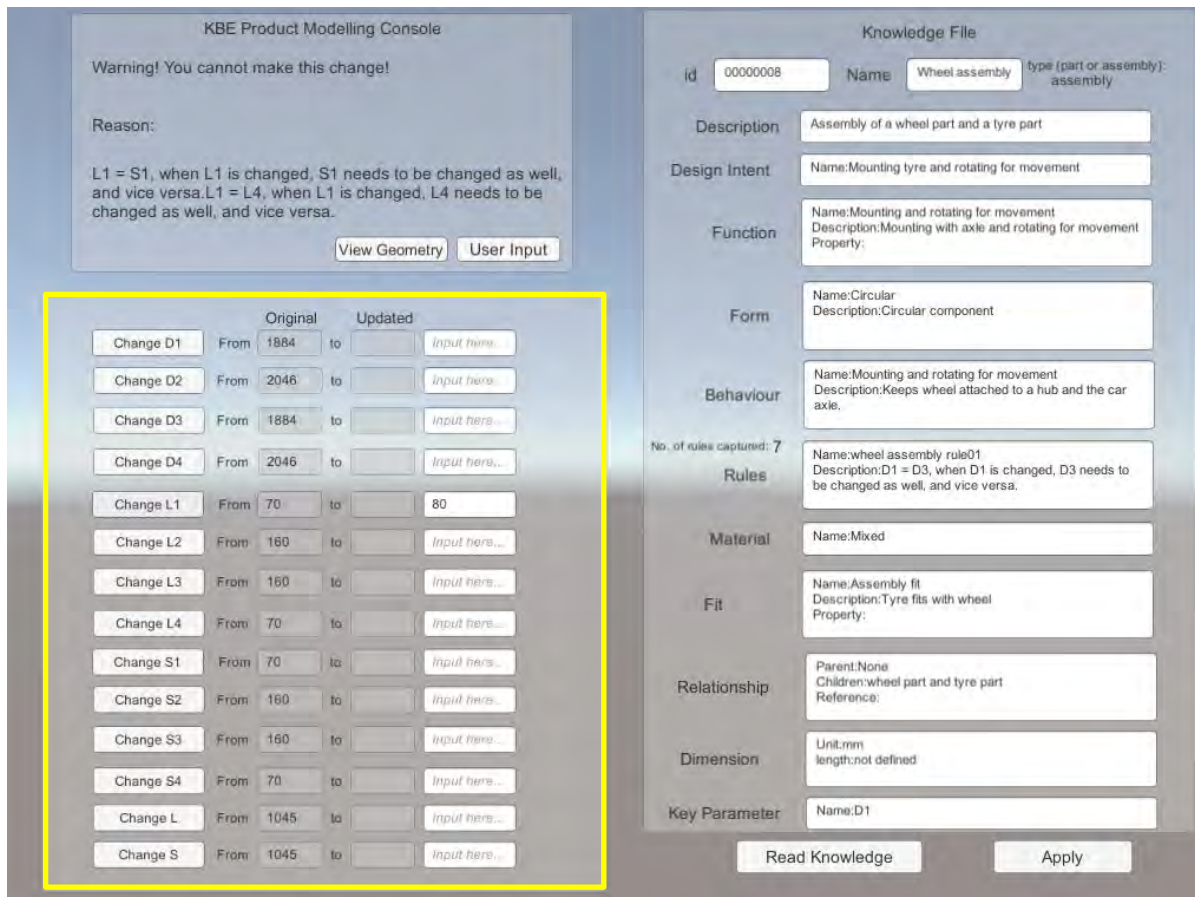


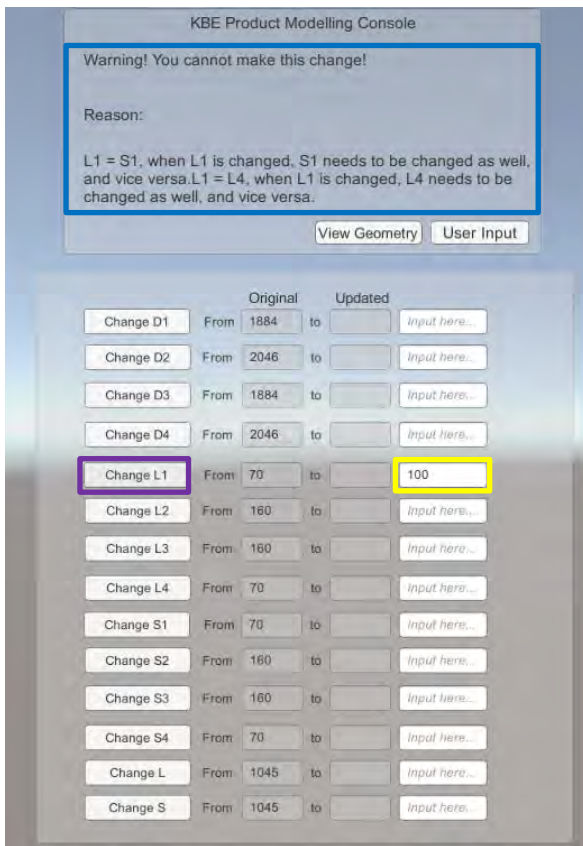
Figure 6-73: Functions of making possible changes to the wheel assembly model in the developed knowledge-based product modelling environment

As described before in Section 6.5.2, two testing scenarios have been defined to verify and validate the tool's effectiveness which are:

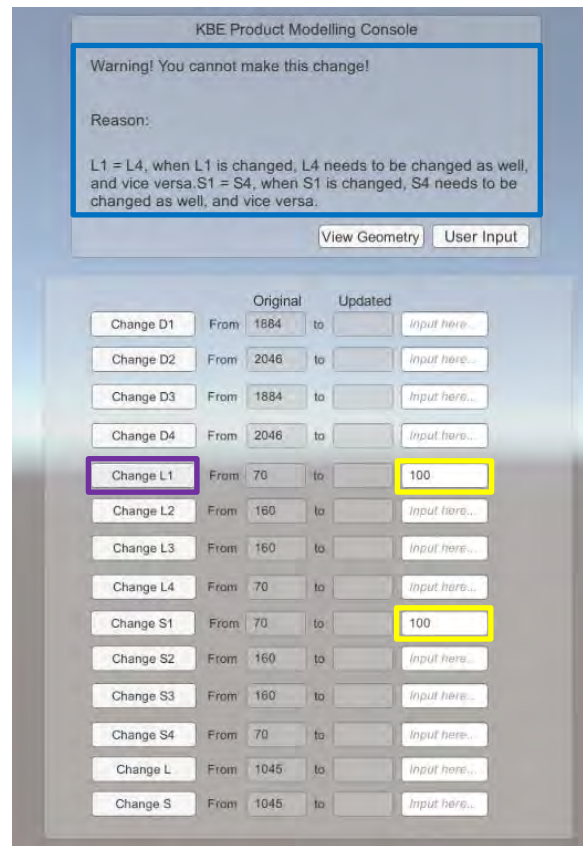
- Scenario One – changing wheel part dimension (with internal constraints from the wheel part itself and external constraints from the tyre part),
- Scenario Two – changing the tyre part dimension (internal constraints from the tyre part itself and external constraints from the wheel).

The validation results of scenario one are shown in Figure 6-74. When the users change the L1 parameter of the wheel part, the tool interface shows that the S1 parameter in the tyre part needs to be adjusted based on the wheel assembly rule 03. In the meantime, the L4 parameter in the wheel part needs to be changed according to the wheel part rule 01, and the S4 needs to

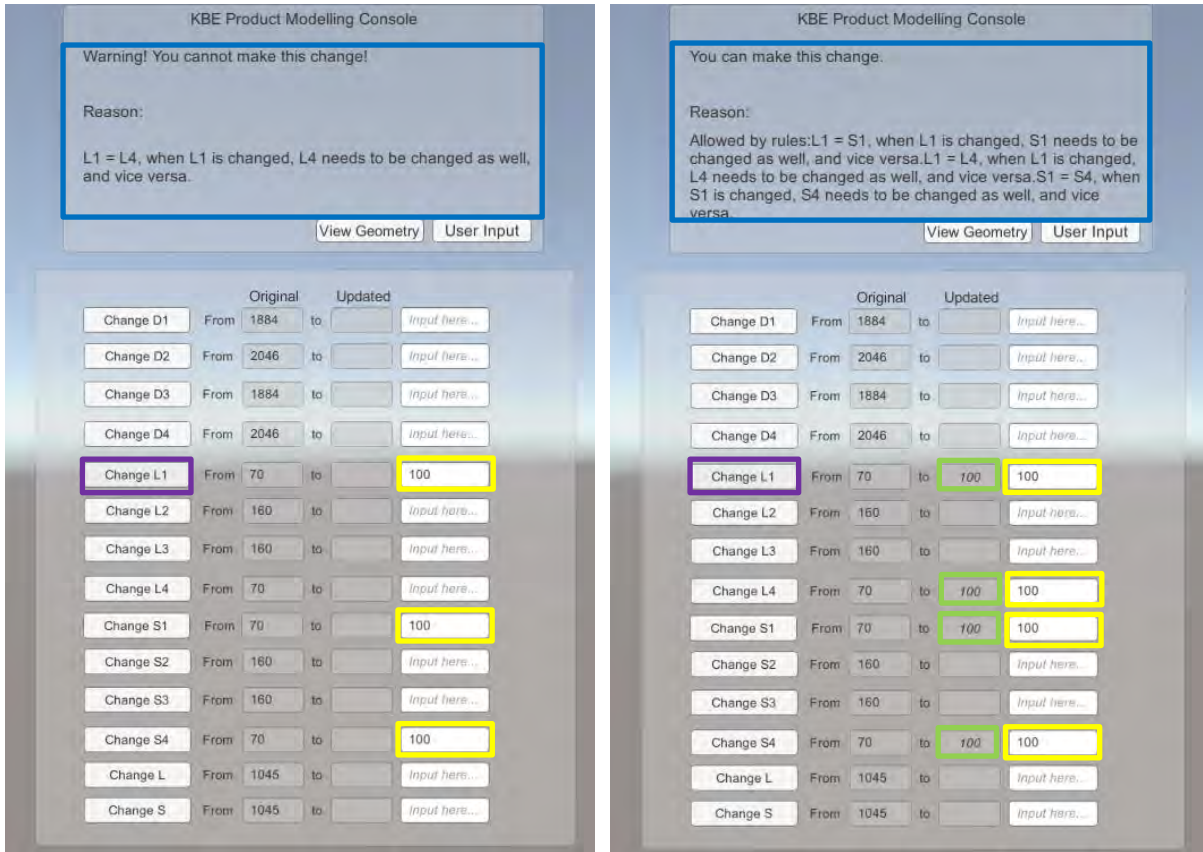
be changed according to the tyre rule 01. Therefore, only when all the required changes are applied will the tool allow the change of the L1 parameter to be made by the users.



(a) input L1 and apply the change



(b) input L1, S1 and apply the change

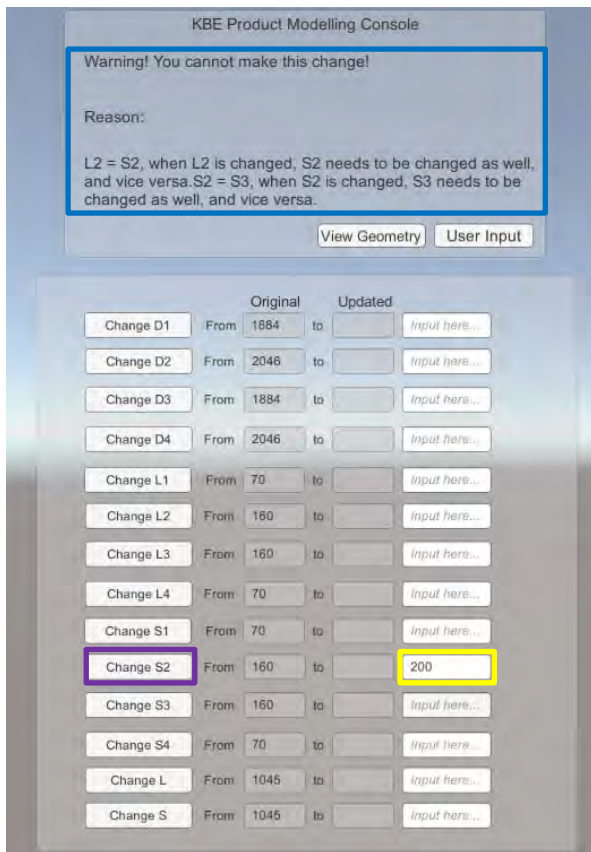


(c) input L1, S1, S4 and apply the change (d) input L1, S1, S4, L4 and apply the change

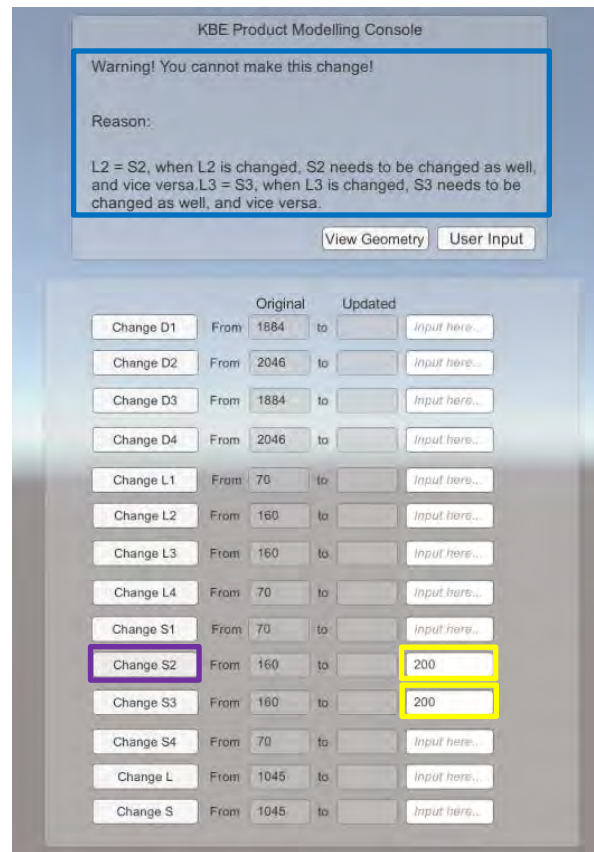
Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box – button pressed to apply the change; Blue box - knowledge reasoning.

Figure 6-74: Results of validation – use case 3: wheel assembly, scenario one - change the wheel part dimension - L1 parameter

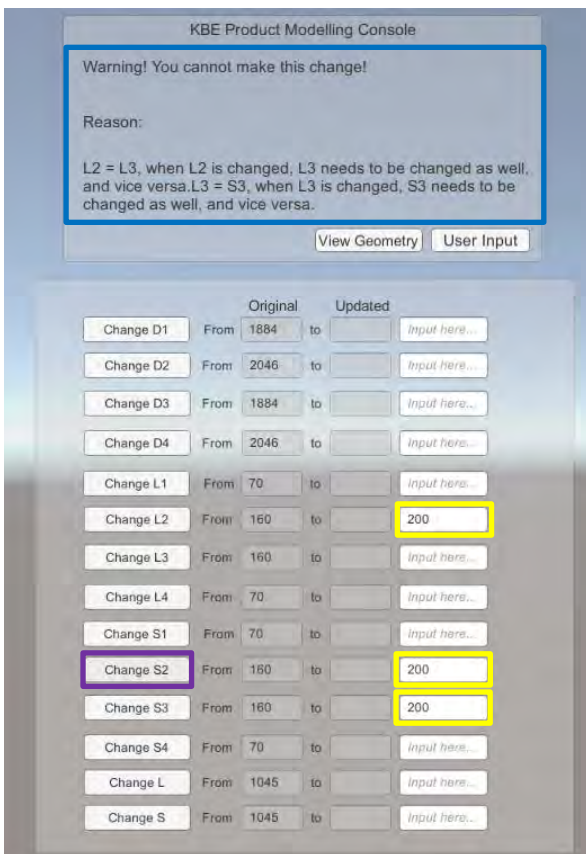
Similarly, the validation of scenario two was carried out, and the results are shown in Figure 6-75. If the users intend to change the S2 parameter in the tyre part, the tool will indicate that this change is constrained by wheel assembly rule 04 in the “Knowledge Product Modelling Console”. Thus, the users need to apply all the required modifications of other parameters before performing the target change of the S2 parameter.



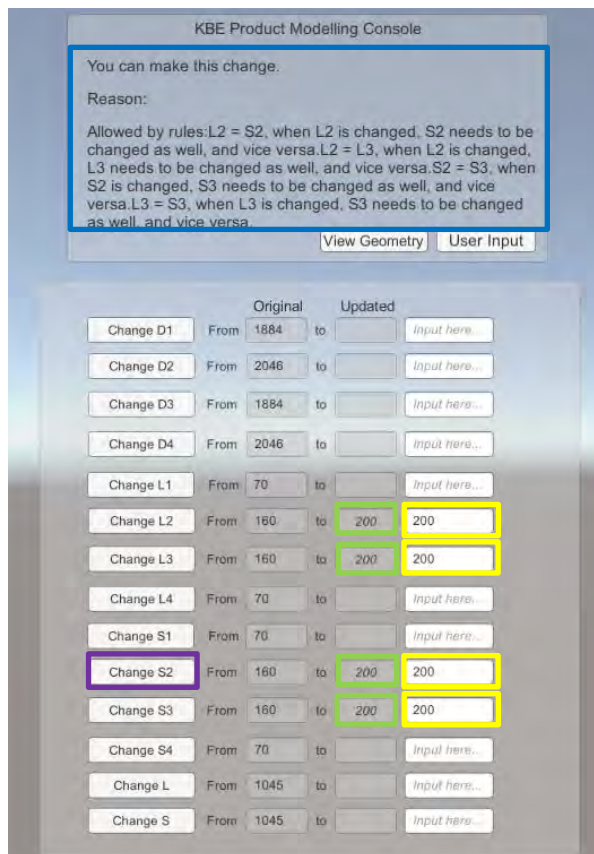
(a) input S2 and apply the change



(b) input S2, S3 and apply the change



(c) input S2, S3, L2 and apply the change



(d) input S2, S3, L2, L3 and apply the change

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box – button pressed to apply the change; Blue box - knowledge reasoning.

Figure 6-75: Results of validation – use case 3: wheel assembly, scenario two – change of the tyre part dimension – S2 parameter

6.5.4 Result Analysis and Use Case Discussion

Assembly organises child parts and part subassemblies to define more complex parts. The deployment of the wheel assembly model aims to evaluate the effectiveness of the framework for implementation with large amounts of product data and complex rules that constrain parameters between parts in one assembly in the product modelling process. Like the previous use cases, the next evaluation objective is to critically analyse the virtual product modelling framework results in use case 3 and then compare the product modelling results from using the virtual product modelling framework with the use of the current existing/legacy product modelling systems for the same circumstances.

The virtual product modelling framework implementation in the wheel assembly example further proves that the framework satisfied all these measurement parameters identified in Section 6.5.2.

1) Generative representation – C1

VPM product model structures are developed for the wheel assembly and its parts to provide a generative representation of the wheel assembly (as shown in Figure 6-56). These developed VPM product model structures have also shown assembly relationships between the assembly and the part. The generalisation of the wheel assembly in both assembly and part levels can help users have a comprehensive understanding of the wheel assembly and its parts and identify the required aspects that should be considered in designing the wheel assembly. Since the wheel assembly parameters have been constrained by both part rules and assembly rules, a variant model of the wheel assembly can be generated quickly by varying

parameters either in the wheel part or in the tyre. This would save the time of creating wheel assembly product variants.

2) Knowledge capture – C2

The visualisation and validation results show that the virtual product modelling methodology is successful in capturing all the existing product information of the wheel assembly use case. Compared with use cases 1 and 2, more complex rules that define internal constraints and external constraints of wheel assembly parameters are captured successfully into knowledge files. All the captured knowledge of the assembly model and individual parts are converted into knowledge files using the knowledge capture tool and represented later in the tool interface. The results of visualisation and validation prove the effectiveness of the developed knowledge-based product modelling environment in capturing complex assembly rules and assembly information from the existing design knowledge.

3) Product geometry and knowledge visualisation – C3

The original geometry and the captured knowledge of the wheel assembly, wheel part and the tyre part are visualised successfully in the interface. Moreover, the changes in the dimension of the wheel part and tyre part are visualised through text description. It proves the capability of the developed knowledge-based product modelling environment in visualising an engineering assembly's geometry and its associated knowledge. Although the visualisation of geometry changes is limited to text, the developed knowledge-based product modelling environment through VPM can still be used to help users realise the complex rules involved in the modelling process of the wheel assembly. It could reduce the time users may spend learning about each assembly component and its connections before starting the modelling process.

4) Product relationship representation – C4

The product relationship representation is achieved during the implementation process. The wheel assembly itself is identified as “assembly” in the developed knowledge-based product modelling environment, and the individual wheel and tyre are identified and shown as “part”. This result proves the correctness of the relationship representation of the wheel assembly example.

5) Knowledge reasoning and reuse – C5

From the implementation results of use case 3, it can be seen that the developed knowledge-based product modelling environment is capable of propagating changes in the parameters of the wheel part and tyre part with knowledge reasoning through text visualisation. All the captured rules, including assembly rules and part rules, are reused effectively in driving and constraining the targeted dimensional parameters of the wheel assembly. Compared with the rules involved in use case 2, the assembly rules and part rules implemented in this use case are more complex. The change of one parameter of the wheel part is restricted by not only internal parameters from the wheel part but also external parameters from the tyre part. As shown from the validation results, a change of S2 in the tyre part will cause the change of S3 in the tyre part and changes of L2 and L3 in the wheel part. Only when S3, L2, and L3 are all correctly set up would the tool allow the users to proceed with the change of S2. The reuse of rules in the developed knowledge-based product modelling environment ensures that all the changes are made correctly according to the assembly relations. And the knowledge reasoning based on the rules provides the users with rationales behind each modelling process that are not allowed by the developed knowledge-based product modelling environment. The resulting knowledge reasoning and reuse in the modelling process of the wheel assembly and its parts further validates that complex rules can be reused for knowledge reasoning and for driving the changes of dimensions of an engineering assembly and its parts through the

application of VPM. Eventually, the knowledge reasoning and reuse for modelling this wheel assembly within the developed knowledge-based product modelling environment would help users save time and reduce errors in changing parameters of the wheel part and the tyre part that are constrained both internally and externally.

6) Correctness of the changes – C6

The correctness of the changes made to the wheel assembly and its parts is proved during visualisation and validation. Due to the lack of enabling tools, the affected changes are shown through text visualisation. By checking the resulted values in the text with the expected results from the pre-defined assembly rules, it can be seen that the changes applied to the geometry of the wheel part and tyre parts are all correct. This guarantees that the wheel assembly are modelled correctly after applying different changes to the parameters.

7) Data exchange – C7 and C8

Data exchange in use case 3 is implemented using STEP files and knowledge files of the wheel assembly and its parts. The exchange of geometry and knowledge is validated by the correct geometry visualisation and knowledge representation. This proves the effectiveness of the proposed data exchange method in VPM in exchanging assembly data by using multiple STEP files and knowledge files for the assembly and its parts, respectively.

The successful implementation of use case 3 further verifies that the VPM meets all the derived measurement parameters from the evaluation criteria. Next, based on the evaluation criteria, a comparison of product modelling results of a wheel assembly between using VPM and using the current existing/legacy CAD systems for the same circumstances are summarised in the Table 6-17.

Table 6-17: Comparison of the use case 3 implementation results between the existing/legacy CAD system and VPM

Evaluation Criteria	Existing/legacy CAD system implementation	Virtual Product Modelling framework implementation
The capability of generative representation of engineering products in VPM	<p>Template model is not available from the library.</p> <p>Need to create the template model separately (geometry representation only and limited to proprietary format).</p> <p>Limited product information is provided.</p>	<p>Develop VPM product model structures as generative representation of the wheel assembly, wheel part and tyre part (in a standardised format - UML)</p> <p>Can provide information, such as function, behaviour, design intent, material, fit, design rules to understand possible product design configurations.</p>
The capability of capturing the product geometry and its associated knowledge from the existing product information	<p>Capture the wheel assembly geometry through importing/exporting the model into standardised format.</p> <p>Unable to capture the existing wheel assembly knowledge during the product modelling process.</p>	<p>Capture the wheel assembly geometry through importing the models from the CAD system in a standardised format.</p> <p>Capture existing knowledge of the wheel assembly during the product modelling process through the knowledge capture tool.</p>
The capability of visualising the product geometry and its associated knowledge	Visualisation of the wheel assembly, wheel part and tyre part (geometry only)	Visualisation of the original geometry of the wheel assembly, wheel part and tyre part and their associated knowledge. Text visualisation of the changes

		of the wheel assembly geometry parameters.
The capability of presenting every part of the product and the relationships among them	Representation of assembly relationships in the hierarchy tree.	The assembly is shown as “assembly”, and the parts are shown as “part” in the proof-of-concept tool interface.
The capability of propagating changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge	<p>Manual tracking of changes of parameters.</p> <p>The change in wheel assembly geometry is reflected in 3D visualisation graphically.</p> <p>No knowledge reasoning is provided when changing parameters in the existing CAD systems. Rules are not available for reusing.</p>	<p>Automatic tracking of changes of parameters through the reuse of rules.</p> <p>Text visualisation of the affected parameters in the proof-of-concept tool interface.</p> <p>The change of wheel assembly geometry is reflected through text description due to the limited available enabling tools.</p> <p>Knowledge reasoning is provided in the proof-of-concept tool interface by reuse of rules from the existing knowledge</p>
The correctness of the changes applied to the product geometry by reuse of the existing knowledge	Manual check-up of the correctness of changes is required.	The changes applied to the wheel assembly geometry follow the rules derived from the theoretical assembling information. The correctness is proved during the validation stage.
The capability and correctness of the product	The geometric data of the wheel assembly, wheel part,	The geometric data of the wheel assembly, wheel part

geometry data exchange between different platforms	and tyre part are correctly exchanged in the format of a STEP file.	and tyre part are exchanged in the format of a STEP file. The correctness is proved during the visualisation process.
The capability and correctness of the knowledge exchange through knowledge file	The existing/legacy CAD systems are not able to exchange knowledge between each other using a generalised format.	The captured knowledge is stored and exchanged in knowledge files in XML. Knowledge files are created separately for wheel assembly, wheel, and tyre parts. Knowledge exchange is proved during the implementation and validation stage.

The critical analysis and comparison further prove the effectiveness of the proposed VPM framework in the chosen assembly use case. Modelling assembly brings more complexity to VPM in terms of parameters, internal and external rule constraints and relations. As the wheel assembly parameters have been constrained by both part rules and assembly rules in the VPM, the generalisation of the wheel assembly using the VPM product model structure allows users to quickly generate a wheel assembly variant model through varying parameters either in the wheel part or in the tyre.

Also, the embedded knowledge in the VPM could help users have a comprehensive understanding of both parent and child levels of the wheel assembly. Users would identify aspects such as function, behaviour, and design intent of each part of the assembly from the corresponding blocks in the VPM product model structure to learn “what the product is

supposed to do”, “how the product implements its function”, and “reasons for making or modifying this product”.

Furthermore, knowledge reasoning and reuse in the modelling process of the wheel assembly in VPM ensure that all changes made to each dimensional parameter of the wheel part and the tyre part are in accordance with assembly rules and part rules. This would help users save time and avoid making mistakes when modelling the wheel assembly and its part. The knowledge exchange using knowledge files provides a formalised way of storing and transferring the captured knowledge among different knowledge-based product modelling environments through a platform-independent XML format.

Based on the results analysis and use case discussion that was carried out for each use case, a further discussion and findings from the use case evaluation are provided in the next section.

6.6 Discussion and Findings

In comparison to the existing/legacy CAD systems, the developed VPM product modelling environment has shown extended capabilities of generative representation, knowledge capturing, reasoning, reuse, and exchange to enhance the product modelling process. The successful implementation of three use cases has proved the effectiveness of VPM in capturing and reusing the existing product information as knowledge to provide a product model structure for the generative representation of a product and knowledge reasoning in the product modelling process. The visualisation of product geometry and its associated knowledge has validated the proposed data exchange method for exchanging the geometric data and knowledge (non-geometric data) of a product.

From the result analysis and use case discussion of the first use case, it can be seen that there is no significant distinction between modelling simple parts by using VPM and using existing/legacy CAD systems due to the simplicity of the product and its associated

knowledge. Design rules involved in the modelling process are uncomplicated, and only a few parameters could be used to vary the model. In this case, the effectiveness of knowledge capture and reuse in the product modelling process through VPM may not be evident because understanding the design rules and modelling these simple parts is straightforward; Users would model the simple parts quickly and correctly even without the support of knowledge reasoning and using the existing product information as guidance.

However, the result analysis and use case discussion of use cases 2 and 3 show that the difference in product modelling between using VPM and existing/legacy CAD systems becomes more evident when the product becomes more complex and has more design rules and parameters. The level of generalisation using VPM depends on the richness of product data. Given adequate existing product information, VPM could be used to develop a higher level of generalisation for engineering parts and assembly. The explicit design knowledge integrated with the developed VPM product model structure could help users such as less-experienced design engineers, multidisciplinary teams involved in the product development process, and non-engineers to understand the product better and identify the essential data according to their design specification. The reuse of design rules that define connections and constraints between different internal and external parameters would allow VPM to generate more product variants with different combinations of parameters in an agile way. The knowledge reasoning and reuse would help users automatically check the correctness of changes according to the captured design rules. This would save the time needed for making changes and eliminate errors during the modelling process. And the knowledge exchange using knowledge files provides a steady data exchange method for storing and transferring the captured knowledge between different knowledge-based product modelling platforms through an interoperable XML format.

Therefore, the enhancement to the product modelling process by using VPM will become more obvious when modelling products with more complexity regarding existing product information, parameters, internal and external rule constraints and relations.

Critical analysis of the implementation results between product modelling in the existing/legacy CAD systems and the proposed framework application proved the advantages of this research and highlighted the capability of knowledge capture and reuse of the developed knowledge-based product modelling environment. These include the following:

- Capture existing associated product knowledge during the product modelling process.
- Visualisation of product geometry and its associated knowledge.
- Automatic tracking of changes of parameters.
- Text visualisation of the changes between the original and modified product models with the affected parameters.
- Knowledge reasoning by reuse of the existing knowledge.
- Knowledge exchange in a platform-independent neutral format for data communication.

To ensure that the Virtual Product Modelling Framework shows full effectiveness, it was essential to compare it against the KBE framework requirements identified in previous Section 3.6.5. These requirements are as follows:

- Generative Modelling – this requirement states that the developed method should provide a generic representation of a product that stores design intent and product configuration information. This requirement was achieved through the use case evaluation that involves the development of a VPM product model structure to represent a generic product model based on the selected use case with all associated information.

- Common Computational Model - this requirement states that the developed framework should provide a data exchange method as a common interface to connect models with associated applications tools. This requirement was achieved through the use of neutral standards to represent product data. This will allow the generated product model from the proposed framework to be interoperable between different applications. The use case evaluation proved that the proposed knowledge exchange method successfully exchanged product knowledge through a knowledge file in the developed knowledge-based product modelling environment.
- Design Optimisation - this requirement states that the developed framework should be able to integrate rules to help identify the best combination of the product performance and driving parameters and avoid making mistakes in engineering tasks. It was shown in the use case implementation that design rules were successfully captured and reused in the developed knowledge-based product modelling environment to provide knowledge reasoning for design engineers. The developed knowledge-based product modelling environment has been proven to contain the necessary capabilities to use rules to identify the constrained parameters and help the design engineering avoid making mistakes during the product modelling process.

This section also compared the proposed VPM framework with existing product models and KBE methodologies to show how VPM is different from them for enhancing the product modelling process. Based on the literature review conducted in Section 2.3.1, it was identified that existing CAD models do not provide enough design information in the product modelling process, and there is a lack of design knowledge representation in existing product models. The Virtual Product Modelling framework addresses this gap by presenting a product model structure in UML for knowledge representation of a product. Previous researchers have tried to develop generic product models and expand the product data representation by

adding textual information. Compared with the Core Product Model developed by Fenves (2001), the proposed Virtual Product Modelling framework provides detailed implementation steps for the application of the presented VPM product model structure. The Core Product Model mainly provides content-level design information for users; in contrast, the developed product model from VPM provides an inference mechanism between the product model geometry and the design information through the use of the captured design rules. This allows the model to be further implemented in the product modelling process. Wang et al. (2003) and Mehmet et al. (2005) extended the Core Product Model (Fenves, 2001) by adding different classifications such as design rationale and assembly relations to enhance the existing product model; however, only text-based implementations were provided in their research. This research adopted some of the concepts from their work to define meta classes and structure the product model. Different from their work, additional visualisation capability was provided to implement the developed model through the use of STEP file. This allows the users to visualise the geometry of the product that is being modelled. In contrast to the work of Jurit H., Saia, A. and De Pennington (1990), where knowledge reasoning techniques are deployed for representing knowledge in the early product planning stage, the proposed framework provides a generic model with the capability of knowledge reasoning in the product modelling stage. This allows the methodology to be more adaptable for design engineers in the product modelling process to avoid making mistakes. Compared to the frame-rule structure used by Lou, Jiang and Ruan (2004), where the rules are represented and used to reason the knowledge in mould-base design, this proposed methodology provides a generic product modelling method that can be used to represent various engineering products. In this proposed method, the product is not limited to a particular type, and a generic VPM product model structure that represents all associated knowledge of the product is developed. In contrast to the work of Salustri (1996), the proposed method provides a product model

along with an adaptable product modelling implementation framework which offers detailed implementation steps for the application use. This allows the developed product model to be further applied and implemented by users for their product modelling purposes. Compared to the work of Gross *et al.* (2009) and Gross and Rudolph (2012), where a unified UML product model is used to integrate domain-specific information of a satellite design, the current research proposed a VPM product model structure that can be used to represent generic products in UML. Gross *et al.* (2009) used UML instances to set the link between UML and the CAD software. However, these UML instances are limited to CAD proprietary native formats. Unlike other researchers' work, this research provided a product model that uses the neutral STEP file to link the geometry of the product model with CAD software and a knowledge file to link the geometry with the stored knowledge. Both files follow interoperable standards and are directly imported into the developed knowledge-based product modelling environment. This allows the developed product model from VPM to be exchanged across different platforms.

Rocca (2011) used DEE methods (explained in Section 3.2.2) to develop a generative aircraft design. In Rocca's work, a complex UML structure was used to represent a generative aircraft model, and all the knowledge is structured and linked with the aircraft model for the purpose of multidisciplinary design optimisation. In contrast to the work of Rocca, this research focused on developing a product modelling methodology that enables the capturing and reusing of existing product knowledge for product modelling to support design engineering automation. Knowledge capturing and exchanging methods are provided in this research, and these methods allow users to capture and transfer the existing design knowledge in the product modelling process.

The literature review of existing product modelling methodologies has identified a lack of knowledge capture and reuse in the product modelling process. Although knowledge-based

engineering techniques provide the capability of capturing and reusing knowledge (Chapman *et al.*, 2007; Rocca, 2012), the existing KBE methodologies show a “black box” problem in understanding KBE applications and the substantiation steps for the implementation of KBE frameworks are still limited. To overcome the “black box” problem in the existing KBE methodologies, the current proposed framework provides a knowledge-based product modelling environment that enables knowledge capture and reuse for the product modelling process. It provides five implementation stages to substantiate this methodology with enabling tools and data exchange methods. This would allow the framework to be adapted and reused by users in accordance with different use cases and different requirements of development. And the developed knowledge exchange method also addressed the need for transferring design engineering knowledge of a product between KBE applications using a formalised file.

The evaluation results also show that the existing/legacy CAD systems are limited in capturing and visualising the existing design knowledge in the product modelling process. Manual tracking of changes of parameters and manual check-ups of the correctness of changes are required, and knowledge reasoning is not provided when changing parameters in the existing/legacy CAD systems. Few CAD systems provide the capability of defining rules and constraining the geometry by using the pre-defined rules, and those rules are limited to simple logic expression and mathematical algorithms. Moreover, the existing/legacy CAD systems are not able to exchange these rules through a generalised format.

Use case evaluation results of the proposed framework show limitations of the developed product modelling environment in the functionality of making changes to product geometry and 3D visualisation of the modified product model. The existing/legacy CAD systems provided better functions in modelling and visualising the product’s geometry through the use of the CAD native formats and mature CAD modelling engine. In the proposed knowledge-

based product modelling environment, the change of geometry was reflected through text description. Modelling functions were limited to use case for making changes to the product models. However, these limitations are acceptable as the system has demonstrated its overall effectiveness in constraining and propagating the changes by reusing the knowledge. The identified limitations are further discussed in Section 7.5.

Based on these findings discussed above, it can be concluded that the results of the virtual product modelling framework implementation are adequate and viable. The proposed methodology has addressed all the identified research gaps by providing a virtual product modelling framework for capturing and reusing existing product knowledge for product modelling to support design engineering automation. Future work and limitations are further discussed in the conclusion chapter.

6.7 Chapter Summary

The developed virtual product modelling framework was applied to three different use cases, and these use cases worked through the implementation framework. Different testing scenarios were distinguished, and three use cases were verified and validated through their application. This chapter presented the evaluation of the effectiveness of the proposed framework. It showed that all the evaluation criteria and measurement parameters had been matched based on the evaluation process through the use cases and the critical analysis between the implementation results in existing/legacy CAD systems and the proposed framework. The strength of product modelling with VPM was reflected in modelling products with more complexity in terms of existing product information, parameters, internal and external rule constraints and relations.

Next, discussion and findings were provided by further analysing and comparing the VPM with existing product models, KBE methodologies and CAD systems. It showed VPM's advantages over the existing product models and KBE methodologies in terms of providing:

(i) an inference mechanism between the product model geometry and the design information through the use of the captured design rules, (ii) an adaptable product modelling implementation framework with detailed implementation steps and enabling tools for the application use, (iii) a data exchange method that allows the geometric data and the captured knowledge of a product model to be exchanged between different platforms, (iv) a knowledge capturing and reuse method, for enhancing the product modelling process. It also showed that the main differentiators of VPM with existing/legacy CAD systems are the capability of generative product representation as well as capturing, reasoning, reuse, and visualising the knowledge in the product modelling process for modelling products with more complexity in terms of existing product information, parameters, internal and external rule constraints and relations.

Based on the evaluation results, the next chapter discusses the research outcomes in more detail, describes the limitation of the current work, provides recommendations for future research, and draws overall conclusions about this research.

7 Conclusion and Recommendation

7.1 Introduction

The research has been successful in its development of the virtual product modelling framework to enhance the product modelling process for design engineering automation in the knowledge-based product modelling environment. This chapter first summarises the work that has been done. Next, the research outcomes are discussed, showing how each research question is answered and how each objective is addressed in this research. Furthermore, this chapter states the contributions to knowledge and discusses the limitations of the study. At last, this chapter provides possible directions for future work, such as product modelling standard development and knowledge-based product modelling application development.

7.2 Summary

The aim of this research is to develop methods and tools for capturing and reusing the existing product knowledge to enhance the product modelling process for design automation in a KBE environment. To achieve that, this research has introduced a novel virtual product modelling approach. As illustrated in Chapter 1, the aims and objectives of this research were derived from the general limitations of traditional product modelling in terms of knowledge utilisation for design engineering automation. The literature review of the current product design and modelling methods in Chapter 2 explained how the product design has evolved with computer-aided technologies since the early 1960s. The review of design engineering automation with product models formed the basis of how automation can be achieved in product modelling to support design engineering by implementing the reuse of knowledge. Two key aspects of design engineering automation should be considered for product modelling: reuse of CAD models and reuse of existing knowledge. Chapter 2 further discussed different product modelling methods existing in the literature for the development

of the product model. It was identified that knowledge-based product modelling emerged as a powerful technique to provide knowledge capturing and reusing capabilities in product development to reduce the time and manpower cost in the design stage. Chapter 2 moved to explore knowledge representation in product model development for knowledge capturing and reusing and discussed the limitations in the product model development. These limitations include a lack of substantiation for the application of the product model through use cases and tools for the purpose of knowledge capturing and reusing in the product modelling process and unclear interaction between the product model geometry and the design engineering automation in the existing product models. Based on the literature, a knowledge reasoning approach with rules was necessary to perform the inference mechanism between users and product models. To take this further for product modelling, it was necessary to use design rules for knowledge reasoning to achieve the inference/interaction between geometry and design information in the product models. From the literature review conducted in this chapter, it can be seen that the product model needs to provide a generative product representation that can provide all associated design information for product modelling and also offer the capability of knowledge reasoning with the captured design rules. Knowledge-Based Engineering is a relatively new enabling method that provides a combination of object-oriented programming, Artificial Intelligence techniques and computer-aided design technologies for knowledge reuse and design automation. Since this research focuses on knowledge-based product modelling methods in application to support design engineering automation, distinguishing different knowledge-based engineering techniques allowed the author to find the most applicable method for capturing and reusing knowledge in product modelling.

Chapter 3 provided a further discussion about different knowledge-based engineering techniques, product modelling standards, and tools to understand better how to capture and

reuse product design knowledge for developing a knowledge-based product modelling framework. A range of KBE techniques have been reviewed, and KCM has shown its advantages in capturing, structuring, and decomposing design knowledge for design automation in product modelling. Studies carried out by researchers showed that KBE systems can be developed to capture the product and process information to support the modelling of engineering or business processes, and the resulted model from KBE systems could be used to automate all or part of the process, which will shorten the development of the product and help to deliver the design faster (Chapman *et al.*, 2007; Rocca, 2012)

Even though KBE methodologies have been successful in dealing with knowledge capture, structuring and reusing, researchers recognised that there are “black-box” problems in the communication between different KBE systems (Cederfeldt, Elgh and Rask, 2006; Fan and Bermell-Garcia, 2008). Performed tasks and processes by the KBE systems are implemented in a way that is not readable and understandable to the end-users. The transparency of KBE systems is necessary to provide adaptable and reusable substantiation steps. Limited implementation advice, examples, use cases with enabling tools and techniques are provided in the existing KBE methodologies (Curran, Verhagen and Van Tooren, 2010).

The literature review of KBE methodologies identified limitations that need to be addressed in this research for the development of a product model. It emphasised the need to develop a KBE implementation framework to capture and reuse design knowledge in product modelling to support design engineering automation. Model-based engineering is an emerging approach that uses models as an integral part of the technical baseline to deal with the increasing complexity of systems. It includes the requirements, analysis, design, implementation, and verification of a capability, system, and product throughout the acquisition life cycle. Since a product model itself can be regarded as a system and the implementation of the KBE framework requires the development of a prototype KBE system (proof-of-concept tool),

Chapter 3 further discussed the development of a product model and the development and implementation of a KBE framework given aspects of model-based engineering. The concepts of applying visual modelling languages (UML/SysML) to represent product model structure and adopting a neutral standard and format to support product model data exchange between different product modelling systems were identified as two key aspects of avoiding the “black-box” problems in KBE system communication. Moreover, Chapter 3 also provided a literature view of the product modelling formats and standards as well as tools. The formats of product models have co-evolved along with the development of CAD software. The literature review identified that neutral product modelling standards are not proprietary and typically used as neutral 3D formats for sharing product data between different CAD software. By comparing IGES, STEP and JT standards, it can be seen that STEP is the most suitable neutral product modelling standard for developing an interoperable product model as it provides steady data exchange and is also widely used in industry and supported by common CAD software. STEP is an internationally recognised standard that provides a uniform data representation and information exchange mechanism used in the product life-cycle. However, based on the literature review, it can be identified that modelling a product with the EXPRESS language is complex and time-consuming. Although EXPRESS-G has been introduced in the literature as a graphical notation of EXPRESS language, it cannot reach the full expressiveness of EXPRESS. There is a lack of possibilities to visualise functional components, local or global rules, and algorithms when modelling product data with EXPRESS and EXPRESS-G (Arnold and Podehl, 1999). STEP has published different application protocols for data exchange in different industrial domains. Among those application protocols, AP242 was claimed to be the latest standard by STEP, which would allow the CAD data exchange and archiving between some different protocols and improve the efficiency of processes by integrating the various enterprise functions.

However, the implementation of AP242 in commercial CAD software is still minimal and unspecific in literature (Coronado, 2014; Schätzle, 2016). Even though many research have been done by utilising STEP to support product modelling in recent years, one obvious limitation of the STEP format is it does not allow for the exchange of parameters, design intent and other data that may be associated with the CAD models (Kim *et al.*, 2008). To address this, some research work have been done to enrich the product model data in STEP by mapping external data with STEP entities and classes; however, mapping between different languages could generate new problems such as data missing and mismatching and the mapping with entire STEP standard was complicated and time-consuming (Barbau *et al.*, 2012). There is still very limited research about how to perform the integration of STEP and product modelling methodologies with existing data resources to generate a completed product modelling framework (Yang *et al.*, 2008). Hence, a need for a new product modelling and implementation method for integrating STEP and various product data was identified.

Chapter 3 also provided a review of a range of product modelling tools to identify the best implementation tools that are capable of displaying the CAD models and capturing and reusing the existing knowledge during the product modelling process. Through the literature review, it was identified that there are very limited tools that support knowledge capture and reuse of the existing design knowledge for product modelling while using a generic product model which consists of all the captured knowledge within an interoperable standard. It can be understood that there is a need for the product modelling software and environment to provide more interaction between end-users and the product modelling process through the reuse of existing knowledge to support the product modelling. Hence, a review of interactive application development was performed to understand what tools can be used as an integrated development environment. Unity was identified as the most appropriate implementation tool for developing a product modelling environment using KBE techniques. The conducted

literature review in Chapter 3 further discussed different aspects that should be considered for design engineering automation using KBE techniques and KBE systems. By reviewing existing frameworks and applications that have been developed for product modelling through the use of KBE techniques in the recent decades, Chapter 3 presented that the limitation of the existing relevant research work lies in the capture of design rules of the product model and there is lack of KBE approaches that focus on capturing, modelling and transferring design knowledge of a product for product modelling in KBE applications.

The outcome of the conducted literature review has identified research gaps (see Section 3.8.2) that need to be addressed in this research and stated the need for knowledge capture and reuse in product modelling. In order to address this need, the current research proposed a virtual product modelling framework. As presented in chapters 4 and 5, it was developed through the selection of appropriate enabling methods and tools and was based on the concepts of product design, product modelling, design engineering automation, and knowledge-based engineering. In the end, the knowledge capture methodology was identified and used for capturing and reusing knowledge, and STEP was selected as the most suitable standard for representing the product geometry. To address the need to transfer design engineering knowledge of a product in KBE applications, a data exchange method is developed as the solution for exchanging non-geometric knowledge between the product modelling environment and the knowledge-based product modelling environment. Further, a knowledge-based product modelling environment was developed as a proof-of-concept tool to show the effectiveness of the developed framework.

The proposed method is evaluated using three use cases from the literature. The full evaluation of the framework is described in Chapter 6. The verification and validation through use cases are based on the data collected from the literature. Four simple parts are selected in the first use case to assess the effectiveness of VPM in application to simple parts

with primitive design features (block, cylinder, sphere, cone) and brief design knowledge. In this use case, simple design rules that constrain single parameters were applied to test the knowledge capture and reuse in VPM for modelling simple parts. The second use case is the hex bolt example as a single basic engineering part. The hex bolt is selected as it is one of the most widely used basic engineering parts in the industry. This use case tests a more complex scenario where design rules are collected from industry standards and constraining internal parameters of a part with different conditions. The third use case selects a wheel assembly as a common engineering assembly to validate the developed VPM framework. A wheel assembly is chosen as it is a crucial part of most automotive and has been widely used in the literature as a demonstrative model to explain model structure, component relationships and complex parameter configurations. In this use case, more complicated rules are applied to evaluate the effectiveness of VPM in application to a product that has assembly relationships and both internal and external parameter constraints.

Results from the use case evaluation showed that the developed knowledge-based product modelling environment using VPM could automatically and correctly interpret and visualise both the geometric data and the captured knowledge (non-geometric data) of the imported product model. The tested product models were successfully represented by the captured product knowledge with decomposed atomic blocks in a UML structure using the derived VPM knowledge classes. Also, the developed knowledge capture tool and the implementation of the developed knowledge exchange methods showcase the framework's capability of allowing the users to capture the existing product knowledge in the product modelling process and store and transfer the knowledge through the use of a knowledge file. It was demonstrated in a series of scenarios that the framework could reuse the captured knowledge to support the product modelling. When changes are made by the users, the framework successfully analyses and reuses the rules from the captured knowledge to

indicate if the changes can be made and also shows the reasons that are constraining the changes. The changes in product model geometry are reflected through text visualisation, and changes in parameters are propagated in the developed knowledge-based product modelling environment correctly.

Discussion and findings from the use case evaluation were also presented in Chapter 6. There is no evident distinction between modelling simple parts between using VPM and using existing/legacy CAD systems due to the simplicity of the product and its associated knowledge. The level of generalisation of a product and the completion of knowledge reasoning and reuse in VPM depends on the richness of product data. Thus, the enhancement to the product modelling process from using VPM will become more significant when modelling products with more complexity in terms of existing product information, parameters, internal and external rule constraints and relations. If there is sufficient product design knowledge, VPM could be applied to develop higher level of generalisation for engineering parts and assembly. The developed product model using VPM could provide users such as less-experienced design engineers, multidisciplinary teams involved in the product development process, as well as non-engineers with explicit essential design knowledge engineers to have a comprehensive understanding of the product and identify the required data based on their design specification. It would help to save time in retrieving product data among different knowledge sources for learning the product. The reuse of design rules that specify relations of parts and constraints between different internal and external parameters would enable a quick generation of product variants using VPM. This would save the time and manpower cost of creating product variants for design engineering automation.

Moreover, the knowledge reasoning and reuse would assist the users in automatically checking if the changes they made are accepted by the captured design rules. It would decrease the time needed for making changes to each parameter separately and avoid making

mistakes in changing product parameters during the modelling process. Furthermore, the knowledge exchange using knowledge files provides a steady data exchange method for storing and transferring the captured knowledge between different knowledge-based product modelling platforms through an interoperable XML format.

Additionally, the proposed framework was discussed against the distinguished KBE framework requirements. The requirements of generative modelling, a common computational model and design optimisation by the KBE framework have been achieved by the successful development of a VPM product model structure, a knowledge exchange method and integration with design rules into the product modelling process during the implementation of VPM. Also, comparisons between VPM and the existing product models, KBE methodologies, and CAD systems in Chapter 6 show that VPM fully fills the research gaps and addresses the need identified from the literature review in this research. As a result, the application of VPM will enhance the product modelling process with the capability of knowledge capturing and reuse and help users save time and manpower costs in the product design stage.

7.3 Research Outcomes

With the support of the framework, the thesis answered the research questions and showed how the existing product design knowledge could be captured and reused for product modelling in a KBE environment to support design engineering automation. Research questions and hypotheses were stated previously in Section 4.2. The following subsections outline how each research question was addressed.

7.3.1 Research Question 1

Research question 1: How can the design knowledge be structured and represented through a product model?

This research question was addressed by developing a product model using the proposed method to provide a generative product representation that can provide all associated design knowledge for product modelling and provide the capability of knowledge reasoning with the captured design rules. This developed product model was structured with knowledge classes derived from previous related research in product modelling. UML was used as a visual modelling language to provide a comprehensive product model structure. The design knowledge was broken down into atomic blocks in this UML structure, which describes the product from different aspects. A knowledge schema was further derived based on the knowledge classes, and a knowledge file would be generated using the knowledge schema with the help of the developed knowledge capture tool to enable the knowledge exchange of the product model.

7.3.2 Research Question 2

Research question 2: How can this product model be implemented in a knowledge-based product modelling environment?

The conducted literature review showed that it was necessary to adopt a neutral standard and format for product model data exchange to avoid the “black-box” problems in the KBE system communication. This research question was first addressed by identifying the appropriate product modelling standards and tools for implementation. STEP was used in this research as the most suitable neutral product modelling standard for developing an interoperable product model because it provides steady data exchange and is also widely used in industry and supported by common CAD software. To overcome the difficulty of integrating STEP with existing knowledge, in this research, STEP was used only to store and transfer the geometric information of the product model. A knowledge exchanging method was provided to enable non-geometric knowledge exchange by the author. The captured non-geometric knowledge is stored and exchanged through the interoperable and platform-

independent format XML. After reviewing the current product modelling tools, it was identified that there are limited tools that support the implementation of the proposed product model. Further, to address this need, a gaming engine – Unity, was selected as the most appropriate implementation tool for developing a knowledge-based product modelling environment as a proof-of-concept tool. This implementation tool was developed by the author using object-oriented programming, and the developed knowledge-based product modelling environment was further tested with three use cases from the literature.

7.3.3 Research Question 3

Research question 3: How can the principles and practice of knowledge-based engineering be applied to capture and reuse the existing design knowledge for product modelling through a knowledge-based product modelling framework?

To answer this research question, different KBE techniques were reviewed, and KCM was identified and then used as the enabling KBE methods for capturing, structuring, and decomposing design knowledge for product modelling in this research. Five implementation steps were presented (see Section 5.1) for capturing and reusing the existing design knowledge for product modelling, and those steps were adapted from the eleven steps of KCM implementation (Terpenney, Strong and Wang, 2000). The literature review also identified that the limitation of the existing relevant research work lies in the capture of design rules of the product model, and there is a lack of KBE approaches that focus on capturing, modelling and transferring design knowledge of a product for product modelling in KBE applications. To address these limitations, this study presented a knowledge capturing method and tool (see Section 5.3.2) in this virtual product product modelling framework as the solution for capturing design rules and a knowledge mapping step (see sections 5.1.4 and 5.3.4) as the solution for reusing design rules for knowledge reasoning. For exchanging non-geometric knowledge between the product modelling environment and knowledge-based

product modelling environment, the research used STEP to transfer the geometric data and developed a knowledge schema for a separated knowledge file that used to transfer the existing design knowledge. All these proposed methods were further tested with three use cases.

7.3.4 Research Question 4

Research question 4: How can this framework be implemented and applied by designers to enhance product modelling?

To address this question, the overall implementation methods were presented by the author (see Section 5.3). This proposed framework was implemented with three testing use cases (see Chapter 6). The evaluation results showed that the captured knowledge was successfully stored, transferred, and reused for product modelling in the developed knowledge-based product modelling environment. When the design engineers changed one dimension of the geometry of the initial product model, the tool would check the rules that determine this geometry and tell the users if they can make the changes along with the reason and constraining the rules. In such a way, design engineers would understand what will be affected if the geometry is changed in this model and the constraints of these changes. This would help them avoid making mistakes in the product modelling process. This proposed framework can be further applied to different product models based on user's need. The applied standard and implementation tools can vary depending on the enabling technologies; however, the fundamental principle and outcome of this methodology – capturing and reusing existing knowledge for product modelling, will remain consistent.

It can be concluded that all the four research questions have been answered fully by the end of this research. The following subsections further discuss the research hypotheses tested in this research.

7.3.5 Research Hypothesis 1

Research hypothesis 1: The proposed methods and tools for product modelling in KBE can improve the knowledge representation of product modelling by:

- Providing a generic knowledge integrated product model which can represent all associated product information, including geometric data from CAD and non-geometric information such as design intent, design parameters, design rules, etc.

The first hypothesis was retained as the developed framework has presented a generative product model that represents all the associated product information and geometric data. The overall virtual product modelling structure was represented using a UML structure. The geometric data from CAD was stored in a STEP file, and the non-geometric information was captured and stored in a knowledge file.

7.3.6 Research Hypothesis 2

Research hypothesis 2: The proposed methods and tools for product modelling in KBE can improve the existing KBE techniques for product modelling to support design engineering automation by:

- Formalising the product model in a neutral format and interoperable standard and generating a new data exchange method for transferring design engineering knowledge of a product in KBE applications.
- Enabling knowledge capture and reuse in the product modelling process through a knowledge-based product modelling framework.
- Developing a KBE application implementation framework for product modelling.
- Providing detailed substantiation steps for the implementation through use cases and tools.

This hypothesis was retained, as the implementation of three use cases has successfully met the derived evaluation criteria and satisfied the defined measurement parameters. The chosen STEP format for the STEP file and XML format for the knowledge file are all neutral and interoperable. The STEP file was used to transfer the geometric data of the product model, and the proposed knowledge exchange method for transferring existing knowledge has been proved effective through the evaluation process. The evaluation results of the three use cases have also shown that the adopted knowledge capture methodology was successful in capturing the existing product knowledge from the knowledge source, and the captured knowledge was reused for product modelling in the developed knowledge-based product modelling framework through the knowledge mapping. Detailed implementation methods were presented in Section 5.3, and substantiation steps for implementation were provided in Chapter 6. Full codification for knowledge mapping in each use case was provided in Appendix. Thus, both research hypotheses were correct and could be regarded as answers to the research questions as well.

7.3.7 Conclusions

The following conclusions are drawn from the findings of this research:

1) Existing CAD systems have limited capability to capture and reuse the existing design knowledge in the modelling process.

The review of existing CAD systems shows that they only provide geometric data within CAD models and provide limited ability to capture and reuse the design knowledge in the product modelling process.

2) The existing product design knowledge can be captured and stored through the use of the developed knowledge capture tool based on KCM.

KCM can be adapted to decompose the captured knowledge and provide associated parametric values for product modelling. The captured knowledge can be decomposed

and stored into a knowledge file in an XML format by using the knowledge capture tool based on KCM.

3) *Design knowledge can be structured and represented through a product model using VPM.*

The existing design knowledge can be decomposed into atomic VPM knowledge classes and then used to develop a generative product model structure using UML. The developed product model structure can help users better understand the product and identify and access the essential data according to their design specifications.

4) *The product model can be exchanged in a knowledge-based product modelling environment through the use of neutral standards and formats.*

STEP standard can be used to represent and exchange the geometric data of the product model in a STEP file. The XML format can be used to store and exchange the non-geometric product data in a knowledge file.

5) *The existing product design knowledge can be captured and reused by using VPM to enhance the product modelling process.*

Rules from the existing product design knowledge can be captured and reused using VPM to create the interaction between the geometry and the knowledge. It can provide knowledge reasoning when users make changes to the product model, preventing them from making mistakes. Captured product knowledge can be visualised along with the product geometry in the developed knowledge-based product modelling environment using VPM. Changes in parameters can be automatically tracked. Changes between the original and modified product models with the affected parameters can be visualised in text.

6) Gaming engine can be used to develop a knowledge-based product modelling environment to provide the capabilities of knowledge capture, reuse, and visualisation for product modelling.

This VPM framework can be built as a knowledge-based product modelling environment using a gaming engine – Unity, as the development tool. The developed knowledge-based product modelling environment shows extended capabilities of knowledge capturing, reusing, reasoning, and visualising in the product modelling process.

7.4 Contribution to Knowledge

This research makes a noteworthy contribution to knowledge in the domain of Product Modelling and Knowledge-Based Engineering, as outlined in the following subsections.

7.4.1 Virtual Product Modelling Framework

Existing product modelling systems are limited regarding capturing, structuring and reusing existing product knowledge for product modelling. A novel virtual product modelling framework is proposed to provide an approach for capturing the existing product knowledge and reusing it in the product modelling process. The presented framework introduces a novel concept for capturing and exchanging product data in product modelling to support design engineering automation. It enhances the product modelling with the capabilities of generative representation, knowledge reusing and provides a simplified way of capturing and exchanging knowledge. It also provides design engineers the capability of knowledge reasoning when they are making changes to product geometry and, therefore, can save time and prevent engineers from making errors in the product modelling process.

7.4.2 Virtual Product Modelling Structure

The proposed methodology provides a generic structure that could represent the product with all associated knowledge. The existing knowledge of the product is decomposed and

represented as atomic blocks in the structure. This results in a clear representation of the complex product knowledge. And the explicit design knowledge integrated with the developed structure could help users such as less-experienced design engineers, multidisciplinary teams involved in the product development process, and non-engineers to understand the product better and identify the essential data according to their design specification. It will also save time on storing, accessing and retrieving the existing product knowledge. This virtual product modelling structure is another significant contributor to the knowledge.

7.4.3 Knowledge Capture and Data Exchange Method

The focus of this research is to contribute to the challenges in capturing and reusing existing product knowledge for product modelling to support design engineering automation. As this requires the capture of the existing product knowledge, this research adopts the knowledge capturing methodology to capture the existing product knowledge. It presents a knowledge schema in an interoperable and platform-independent format. The captured knowledge will be stored in a knowledge file based on the knowledge schema. This approach allows the captured knowledge to be transferred and exchanged across different platforms. This knowledge capture and exchange approach can act as a useful tool for transferring non-geometric information of a product as knowledge between different KBE applications. It provides well-defined knowledge classes and a formalised method for individuals, enterprises and industries to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats.

To overcome the limitations in the existing STEP standards, the product geometry is exchanged through the use of STEP file, and the non-geometric knowledge is exchanged through the use of the knowledge file in this research. This work may be regarded as a step

toward developing a holistic product model that consists of interoperable geometry data and reusable non-geometric knowledge classes.

7.4.4 Substantiation for KBE System Implementation

The proposed methodology presents a transparent KBE implementation framework where the knowledge is adaptable, structured and reusable by others. It provides substantiation procedures of how KBE systems can be implemented for product modelling. It incorporates a schematic way of capturing and reusing knowledge and a knowledge mapping method through the use of object-oriented programming to perform knowledge reasoning with engineering rules. These novel methods allow the framework to be flexible so that users can make necessary adjustments or adapt them for further development needs. Therefore, this proposed methodology can be viewed as a possible enhancement of existing KBE methodologies and potential guidance for developing and implementing KBE systems for product modelling.

7.5 Limitations of Research

The research addresses the aims and objectives and answers the research questions set in Chapter 1. However, there are still some limitations to this work that are not considered.

The developed knowledge-based product modelling environment has shown the capability of changing the dimensional parameters of product models. However, those changes are limited to use case. The functionality of modelling geometry in the developed knowledge-based product modelling environment is limited. Due to the nature of STEP, there are no existing technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. In this current knowledge-based product modelling environment, geometry changes mainly rely on text visualisation and knowledge description. But this will not restrict the capability of the proposed modelling

method. This limitation is acceptable as the system has demonstrated its overall effectiveness in propagating the changes by reusing the knowledge. New standards and advanced geometry editing tools will accelerate the formation of a well-developed product modelling environment, and the 3D visualisation of the product can be further improved with enabling standards and tools in the future.

Secondly, the developed tool needs to be further developed to become more generic regarding changing the different aspects of geometry for different use cases. However, this would move the tools to the development of a much larger scale product modelling software, and that would require more time and manpower involvement. Since the currently developed tool has proved the effectiveness of this methodology in capturing and reusing knowledge and reflecting changes made by users, developing such a large-scale software is not the focus of this research.

Thirdly, design engineers need to spend extra time using the knowledge capture tool, compared with modelling in a traditional product modelling environment (CAD). This mainly happens at the first time when designers start to model without having a knowledge file. However, it is necessary to spend such time capturing designer knowledge and building the knowledge repository.

Fourthly, in this research, the author has developed a knowledge capture tool for capturing knowledge and a knowledge-based product modelling environment for reusing knowledge for product modelling as proof-of-concept tools. The knowledge capture tool is not integrated within the knowledge-based product modelling environment, and the use of the knowledge-based product modelling environment requires manually exporting and importing the STEP file and knowledge file. However, this research focused on providing the capability of capturing and reusing the existing knowledge for product modelling. The integration of tools and automatic exporting and importing of files can be recommended for future work.

Lastly, the framework has provided a knowledge mapping method to achieve the knowledge reasoning in the product modelling process. This procedure is done through the object-oriented programming algorithm, which links the rules and the dimensional geometry. In the current developed knowledge-based product modelling environment, the rules are written manually into the object-oriented programming algorithm separately. However, this is due to the limit of enabling tools. It can be argued that by using a product modelling environment that can allow users to embed rules in the product modelling process, this limitation could be overcome.

7.6 Recommendations for Future Work

Based on the findings of the research, several recommendations can be derived for future work. These can be summarised as follows.

7.6.1 Improving the Knowledge-Based Product Modelling Tool

As mentioned in the previous section, the developed knowledge-based product modelling environment is limited in changing the geometry of the product models. Therefore, one of the potential future work directions is to further improve and develop the tool and make it more generic and be able to change any kind of product model geometry. Moreover, the concept of using rules to provide knowledge reasoning for product modelling shows another possible direction to improve the current product modelling legacy system. As mentioned in Chapter 3, there are limited tools that provide the capability of capturing and reusing existing product knowledge for product modelling. Expanding this research outcome into the current CAD product modelling legacy systems will help the development of a more functional and powerful knowledge-based product modelling engine. The visualisation functionality of the product model will also be extended through the use of a mature and commercialised CAD engine.

7.6.2 Enhancing Data Exchange in Product Modelling with New Product Modelling Standard AP242

Another possible direction for future work is to use the latest STEP AP242 standard (*STEP AP242 Project*, 2014) to enhance data exchange in product modelling. The new AP242 was released in 2014 to support the current STEP for exchanging product data; however, AP242 functionalities are not developed to the implementation level in commercial CAD software up-to-date. Therefore, in the future, with the development of A242 functionalities in current commercial CAD software, AP242 can be potentially applied to exchange all classified data by STEP for the application of the knowledge-based product modelling framework.

7.6.3 Extending the Product Modelling Standards for Knowledge Capture and Reuse Implementation

This research work also shows future work directions for current international organisations for standardisation to extend their existing product modelling standards for knowledge capture and reuse implementation. As mentioned in Chapter 3, STEP is used to exchange information, which is the outcome of design activities, rather than for the information generated and used through the development of a product. It is limited in exchanging parameters, design intent and other data that may be associated with the CAD models. Some efforts were made to combine STEP with non-STEP product data (Barbau *et al.*, 2012). However, mapping with the entire STEP standard is complicated and time-consuming. Data missing and mismatching still exist in the current mapping methods. Therefore, another potential future direction can be to apply the knowledge-based product modelling framework to develop a product modelling standard that aims at the implementation of knowledge capturing and reusing. This will provide a structured data format to integrate geometric and non-geometric data for knowledge-based product modelling and a seamless way to exchange product models as all data are stored in one standardised file.

7.6.4 Implementing the Framework for Non-Engineers

The findings of the research reveal that the proposed framework is capable of capturing and reusing the existing product knowledge for product modelling to support design engineering automation. Since the knowledge would be captured from experienced designers and be accessible to new designers through the implementation of the framework, this methodology can be considered to support non-engineers who lack the engineering background knowledge to understand and learn product modelling. The developed KBE product modelling environment has the potential of helping non-engineers to carry out design tasks in a more user-friendly way. The captured knowledge in the proposed framework can also be used as a knowledge repository that provides available and accessible product knowledge for non-engineers to thoroughly understand the purpose of each design detail in the existing CAD models and to unravel the complex design references created by other designers.

7.6.5 KBE Application Development Using a Gaming Engine

This research developed a knowledge-based product modelling environment as a proof-of-concept tool through the use of a gaming engine. It has shown the potential of deploying gaming engines to develop a knowledge-based product modelling platform as a KBE application. In the existing CAD systems, most of the modelling process is performed through the interaction with system GUI. However, developing a KBE application is 90% about writing code and 10% interacting with GUI (Rocca, 2011). As mentioned in Section 3.5.3, gaming engines provide an integrated development environment for building interactive application with GUI and programming the backend. They enable users to develop cross-platform applications with customised interface that driven by object-oriented programming codes. In recent decades, gaming engines have been adopted to build industrial applications in various domains (Juliani *et al.*, 2018; Hussain *et al.*, 2020; Unity

Technologies, 2021). Therefore, gaming engines can possibly be used by researchers to develop KBE applications in the future.

7.6.6 Product Modelling with Virtual Reality and Augmented Reality Technology

Virtual reality (VR) and augmented reality (AR) are new types of visualisation technologies that have been rising rapidly in popularities across different domains and professions. VR replaces the real-life surrounding environment through computer-generated signals (sight, sound, touch, etc.), and AR augments the real-life surrounding by overlaying virtual elements on the live view of the real world. With the utilisation of headsets, glasses, controllers and sensors, these technologies provided an immersive way of visualisation and making interaction with virtual and real environments. Some recent research work offers the potential for the use of VR and AR technologies in the field of engineering design and modelling (Huerta *et al.*, 2019; Memarsadeghi and Varshney, 2020). Thus, the virtual product modelling framework can possibly be applied together with VR and AR applications to enable design engineers to visualise the product and perform the product modelling process in new ways.

Reference

- Abramovici, M., Gerhard, D. and Langenberg, L. (1997) 'Application of PDM technology for Product Life Cycle Management', in *Life Cycle Networks*. Springer, Boston, MA, pp. 17–31.
- Al-ashaab, A. *et al.* (2012) 'Knowledge-based environment to support product design validation', *Knowledge-Based Systems*, 26, pp. 48–60. doi:10.1016/j.knosys.2011.06.019.
- Alan Hevner, Jinsoo Park, S.T.M. (2004) 'Design Science in Information Systems Research', *MIS Quarterly*, 28(1), pp. 75–105.
- Alavudeen, A. and Venkateshwaran, N. (2008) *Computer Integrated Manufacturing*. PHI Learning Pvt. Ltd.
- Ap242.org (2016) *AP 242 – Why the convergence*. Available at: <http://www.ap242.org/why-ap242.jsessionid=6bfl16b5b3d7312345d84445206df> (Accessed: 1 September 2016).
- Arnold, F. and Podehl, G. (1999) 'Best of both worlds - A mapping from EXPRESS-G to UML', *The Unified Modeling Language. «UML»'98: Beyond the Notation*, pp. 49–63. doi:10.1007/978-3-540-48480-6_5.
- Arora, S.K. (2021) *Unity vs Unreal Engine: Which Game Engine Should You Choose?* Available at: <https://hackr.io/blog/unity-vs-unreal-engine> (Accessed: 6 December 2021).
- Avvaru, V.S. *et al.* (2020) 'Integration of PLM, MES and ERP Systems to Optimize the Engineering, Production and Business', in Nyffenegger, F. *et al.* (eds) *Product Lifecycle Management Enabling Smart X*. Cham: Springer International Publishing, pp. 70–82.
- Bajaj, M. (2008) *KNOWLEDGE COMPOSITION METHODOLOGY FOR EFFECTIVE ANALYSIS PROBLEM FORMULATION IN SIMULATION - BASED DESIGN*. Georgia Institute of Technology.
- Bajan, M. *et al.* (2011) 'Satellites to Supply Chains , Energy to Finance — SLIM for Model-Based Systems Engineering Part 2: Applications of SLIM', *INCOSE International Symposium*, (June), pp. 20–23.
- Barbau, R. *et al.* (2012) 'OntoSTEP: Enriching product model data using ontologies', *Computer-Aided Design*, 44, pp. 575–590. doi:10.1016/j.cad.2012.01.008.
- Barnes, M. (2007) *Introduction to Collada*. Available at: <https://www.gamedeveloper.com/art/introduction-to-collada> (Accessed: 6 December 2021).

- Berends, J.P.T.J., Van Tooren, M.J.L. and Schut, E.J. (2008) ‘Design and implementation of a new generation multi-agent task environment framework’, *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference* [Preprint], (April). doi:10.2514/6.2008-2142.
- Bhandarkar, M.P. *et al.* (2000) ‘Migrating from IGES to STEP: One to one translation of IGES drawing to STEP drafting data’, *Computers in Industry*, 41(3), pp. 261–277. doi:10.1016/S0166-3615(99)00052-4.
- Blessing, L. and Wallace, K. (1998) ‘Supporting the Knowledge Life-Cycle’, in *Proceedings of the IFIP TC5 WG5.2 Third Workshop on Knowledge Intensive CAD*, pp. 21–38. doi:10.1007/978-0-387-35582-5.
- Bondar, S. *et al.* (2015) ‘Advances in parameterized CAD feature translation’, *Advances in Transdisciplinary Engineering*, 2(July), pp. 615–624. doi:10.3233/978-1-61499-544-9-615.
- Bonnie E. John and Mashyna, M.M. (1997) ‘Evaluating a multimedia authoring tool’, *Journal of the American Society for Information Science*, 48(11), pp. 1004–1022.
- Bouhaddou, I. *et al.* (2012) ‘PLM (Product Lifecycle Management) Model for Supply Chain Optimization’, in Rivest, L., Bouras, A., and Louhichi, B. (eds) *Product Lifecycle Management. Towards Knowledge-Rich Enterprises*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 134–146.
- Boy, J. *et al.* (2015) ‘Recommended Practices for AP242 Business Object Model XML Assembly Structure’, *CAX-IF Recommended Practices*, pp. 1–184.
- Bozdoc, M. (2003) *Introducing CAM*. Available at: <http://mbinfo.mbdesign.net/CAM-Intro.htm> (Accessed: 1 November 2019).
- Brown, A.S. (2020) *7 Biggest Trends for Engineering in the 2020s - ASME*, *ASME.org*. Available at: <https://www.asme.org/topics-resources/content/7-biggest-trends-for-engineering-in-the-2020s> (Accessed: 10 August 2020).
- Butorina, I. V. and Vasilieva, V.N. (2018) ‘Surface modeling in AutoCAD for architectural and structural design’, *IOP Conference Series: Materials Science and Engineering*, 451(1). doi:10.1088/1757-899X/451/1/012125.
- Cam, C.A.D. *et al.* (1983) ‘Ship Hulls , B-Spline Surfaces ’, (December), pp. 37–45.
- Camba, J.D., Contero, M. and Company, P. (2016) ‘Parametric CAD modeling: An analysis

of strategies for design reusability’, *CAD Computer Aided Design*, 74, pp. 18–31. doi:10.1016/j.cad.2016.01.003.

Carlson, W.E. (2017) *Computer Graphics and Computer Animation: A Retrospective Overview*. The Ohio State University.

Cederfeldt, M. and Elgh, F. (2005) ‘DESIGN AUTOMATION IN SMEs – CURRENT STATE , POTENTIAL , NEED AND REQUIREMENTS’, pp. 1–15.

Cederfeldt, M., Elgh, F. and Rask, I. (2006) ‘A Transparent Design System for Iterative Product Development’, *Journal of Computing and Information Science in Engineering*, 6, pp. 300–307.

Chang, K.-H. (2015) ‘Solid Modeling’, in *e-Design*. Academic Press, pp. 125–167. doi:https://doi.org/10.1016/C2009-0-63076-2.

Chang, K.-H. (2016) *e-Design: Computer-Aided Engineering Design*. revised. Academic Press.

Chapman, C. *et al.* (2007) ‘Utilising enterprise knowledge with knowledge-based engineering’, *International Journal of Computer Applications in Technology*, 28(2–3), pp. 169–179. doi:10.1504/IJCAT.2007.013354.

Chapman, C.B. and Pinfold, M. (1999) ‘Design engineering - a need to rethink the solution using knowledge based engineering’, *Knowledge-Based Systems*, 12(5–6), pp. 257–267. doi:10.1016/S0950-7051(99)00013-1.

Chapman, C.B. and Pinfold, M. (2001) ‘The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure’, *Advances in Engineering Software*, 32(12), pp. 903–912. doi:10.1016/S0965-9978(01)00041-2.

Chen, Y.-M. and Wei, C.-L. (1997) ‘Computer-aided feature-based design for net shape manufacturing’, *Computer Integrated Manufacturing Systems*, 10(2), pp. 147–164. doi:10.1016/S0951-5240(97)00006-2.

Chiang, A.T.A., Trappey, A.J.C. and Ku, C.C. (2004) ‘Using Knowledge-Based Intelligent Reasoning To Support Dynamic Collaborative Design’, in *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference 2004*, pp. 1–12.

Cho, J. *et al.* (2016) ‘KBE-PLM Integration Schema for Engineering Knowledge Re-use and Design Automation’, (September 2017). doi:10.1007/978-3-319-54660-5.

- Chungoora, N. *et al.* (2013) 'A model-driven ontology approach for manufacturing system interoperability and knowledge sharing', *Computers in Industry*, 64(4), pp. 392–401. doi:10.1016/j.compind.2013.01.003.
- Coons, S. and Mann, R. (1960) *Computer-aided design related to the engineering design process / 8436-TM-5*. Cambridge, M.I.T. Electronic Systems Laboratory.
- Cooper, S., Fan, I. and Li, G. (2001) 'Achieving Competitive Advantage Through Knowledge-Based Engineering A Best Practice Guide', p. 21.
- Coronado, J. (2014) *STEP standard support in Creo*. Available at: http://www.asd-ssg.org/c/document_library/get_file?uuid=f11b5a3a-5594-4f79-90be-384f452ac94f&groupId=11317 (Accessed: 6 December 2021).
- Curran, R. *et al.* (2010) 'A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD', *Expert Systems with Applications*, 37(11), pp. 7336–7350. doi:10.1016/j.eswa.2010.04.027.
- Curran, R. *et al.* (2015) *Transdisciplinary Lifecycle Analysis of Systems, Proceedings of the 22nd ISPE Inc. International Conference on Concurrent Engineering*. ISO Press BV.
- Curran, R., Verhagen, W.J.C. and Van Tooren, M.J.L. (2010) 'The KNOMAD methodology for integration of multi-disciplinary engineering knowledge within aerospace production', *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, pp. 1–16.
- David, T. (2019) *An Automated Method Mapping Parametric Features Between Computer Aided Design Software*. BRUNEL UNIVERSITY.
- Estefan, J.A. (2008) *Survey of Model-Based Systems Engineering (MBSE) Methodologies*. doi:10.1109/35.295942.
- Fan, I.-S. and Bermell-Garcia, P. (2008) 'International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management', *Concurrent Engineering*, 16(4), pp. 271–277. doi:10.1177/1063293X08100027.
- Fenves, S.J. (2001) 'A core product model for representing design information', *Technical Report No. NISTIR 6736, National Institute of Standards and Technology* [Preprint]. Available at: <http://www.mel.nist.gov/msidlibrary/doc/ir6736.pdf>.
- Fenves, S.J. *et al.* (2004) 'CPM 2: A Revised core product model for representing design

information’, *Nistir 7185* [Preprint].

Fernández-Godino, M.G. *et al.* (2016) ‘Review of multi-fidelity models’. doi:10.1016/j.jcp.2015.01.034.

Fortineau, V., Paviot, T. and Lamouri, S. (2013) ‘Improving the interoperability of industrial information systems with description logic-based models-The state of the art’, *Computers in Industry*, 64(4), pp. 363–375. doi:10.1016/j.compind.2013.01.001.

Friedenthal, S., Griego, R. and Sampson, M. (2007) ‘INCOSE model based systems engineering (MBSE) initiative’, *INCOSE 2007 Symposium* [Preprint]. Available at: http://www.incose.org/enchantment/docs/07docs/07jul_4mbseroadmap.pdf.

Fröhlich, A. (2011) ‘3D Formats in the Field of Engineering- a Comparison’, *ProStep*, pp. 1–24.

Gao, J.X. *et al.* (2003) ‘Application of product data management technologies for enterprise integration’, *International Journal of Computer Integrated Manufacturing*, 16(7–8), pp. 491–500. doi:10.1080/0951192031000115813.

Gerhard Pahl and Wolfgang Beitz (1988) *Engineering Design, A systematic Approach*. Edited by K. Wallace. London: The Design Council.

Gilkinson, N. *et al.* (2015) ‘Building information modelling: the tide is turning’, *Proceedings of the ICE - Structures and Buildings*, 168(2), pp. 81–93. doi:10.1680/stbu.12.00045.

Giovannini, A. *et al.* (2012) ‘Ontology-based system for supporting manufacturing sustainability’, *Annual Reviews in Control*, 36(2), pp. 309–317. doi:10.1016/j.arcontrol.2012.09.012.

Gómez de Silva Garza, A. and Maher, M. Lou (2000) ‘Knowledge Modeling in Design - the MOKA framework’, *Artificial Intelligence in Design '00*, pp. 77–102. doi:10.1007/978-94-011-4154-3.

Gross, J. *et al.* (2009) ‘An Executable Unified Product Model Based on UML to Support Satellite Design’, in *Proceedings of the AIAA SPACE Conference*. doi:10.2514/6.2009-6642.

Gross, J. and Rudolph, S. (2012) ‘Generating simulation models from UML - A FireSat example’, in *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, pp. 1–8.

Gu, P. and Chan, K. (1995) ‘Product modelling using step’, *Computer-Aided Design*, 27(3),

pp. 163–179. doi:10.1016/0010-4485(95)95867-E.

Gujarathi, G P and Ma, Y. (2011) ‘Parametric CAD / CAE integration using a common data model’, 30, pp. 118–132. doi:10.1016/j.jmsy.2011.01.002.

Gujarathi, G. P. and Ma, Y.S. (2011) ‘Parametric CAD/CAE integration using a common data model’, *Journal of Manufacturing Systems*, 30(3), pp. 118–132. doi:10.1016/j.jmsy.2011.01.002.

Haas, J. (2014) ‘A History of the Unity Game Engine for An Interactive Qualifying Project’, p. 44.

Hale, R. (2002) ‘Knowledge-Based Software Systems for Composite Design, Analysis and Manufacturing’, *SAE Technical Paper Series* [Preprint], (724). Available at: <http://papers.sae.org/2002-01-1536>.

El Hani, M.A., Rivest, L. and Maranzana, R. (2012) ‘Product data reuse in product development: A practitioner’s perspective’, *IFIP Advances in Information and Communication Technology*, 388 AICT, pp. 243–256. doi:10.1007/978-3-642-35758-9_21.

Haynes, S.R. and Skattebo, A.L. (2004) ‘Situating Evaluation in Scenarios of Use’, (May 2014). doi:10.1145/1031607.1031624.

Herschel, R.T., Nemati, H. and Steiger, D. (2001) ‘Tacit to explicit knowledge conversion: Knowledge exchange protocols’, *Journal of Knowledge Management*, 5(1), pp. 107–116. doi:10.1108/13673270110384455.

Hevner, A.R. *et al.* (2004) ‘Design Science in Information Systems Research’, 28(1), pp. 75–105.

Van Holland, W. and Bronsvort, W.F. (2000) ‘Assembly features in modeling and planning’, *Robotics and Computer-Integrated Manufacturing*, 16(4), pp. 277–294. doi:10.1016/S0736-5845(00)00014-4.

Hornbæk, K. *et al.* (2007) ‘Use Case Evaluation (UCE): A Method for Early Usability Evaluation in Software Development’, in *Human-Computer Interaction – INTERACT 2007*, pp. 578–591.

Huerta, O. *et al.* (2019) ‘Application of VR and AR Tools for Technical Drawing Education’, (May), pp. 363–366. doi:10.14733/cadconfp.2019.363-366.

Hussain, A. *et al.* (2020) ‘Unity Game Development Engine : A Technical Survey’,

University of Sindh Journal of Information and Communication Technology (USJICT), 4(2), pp. 73–81.

International Council on Systems Engineering (2007) ‘Systems Engineering Vision 2020’, *INCOSE-TP-2004*. Available at: http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.

Isaksson, O. *et al.* (2000) ‘Trends in Product Modelling - an ENDREA perspective’, in *Proceedings Product Models 2000*. Linköping, Sweden.

Isaksson, O. (2003) ‘A generative modeling approach to engineering design’, in *Proceedings of ICED 03*.

ISO/PAS 17506 (2012) *Industrial automation systems and integration — COLLADA digital asset schema specification for 3D visualization of industrial data*. Available at: <https://www.iso.org/standard/59902.html> (Accessed: 6 December 2021).

ISO (1994) ‘ISO 10303-11:1994 The EXPRESS language reference manual’. Available at: <https://www.iso.org/standard/18348.html>.

ISO 10303-1:1994 (1994) ‘Product data representation and exchange - Part 1: Overview and fundamental principles’.

Johnson, A. and Gibson, A. (2014) ‘Chapter 4 - The Tools of the Design Process and Management of Design’, in *Sustainability in Engineering Design*. Academic Press, pp. 113–180.

Jon Holt, S.P. (2013) *SysML for Systems Engineering: A Model-based Approach*. Institution of Engineering and Technology.

Juliani, A. *et al.* (2018) ‘Unity: A General Platform for Intelligent Agents’, (February). Available at: <http://arxiv.org/abs/1809.02627>.

Jurit H., Saia, A. and De Pennington, A. (1990) ‘Reasoning about machining operations using feature-based models’, *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 28, pp. 153–171.

Kahn, H. *et al.* (2001) ‘A generic framework for transforming EXPRESS information models’, *CAD Computer Aided Design*, 33(7), pp. 501–510. doi:10.1016/S0010-4485(01)00049-5.

Kalkan, E., Okur, F.Y. and Altunışık, A.C. (2018) ‘Applications and usability of parametric

modeling', *Journal of Construction Engineering, Management & Innovation*, 1(3). doi:10.31462/jcemi.2018.03139146.

Kc Morris, A.B.F. (1999) *STEP, the grand experience*. Edited by S.J. Kemmerer. Gaithersburg: National Institute of Standards and Technology.

Kedar, S.D. *et al.* (2018) 'Design And Drawing Automation Using Cad Model Application Programming Interface And KBE System: A Review Paper', *International Journal of Advance Research in Science and Engineering*, 07(02), pp. 611–629.

Kellie, A.C. (2010) *The Wireframe Model — Showing 3D Structure with Open Space*.

Khronos Group (2004) *COLLADA Overview*. Available at: <https://www.khronos.org/api/collada> (Accessed: 6 December 2021).

Kim, J. *et al.* (2008) 'Standardized data exchange of CAD models with design intent', 40, pp. 760–777. doi:10.1016/j.cad.2007.06.014.

Klein, R. (2009) 'Knowledge Modeling in Design - the MOKA framework'. doi:10.1007/978-94-011-4154-3.

Könst, J.S., La Fontaine, J.P. and Hoogeboom, M.G.R. (2009) *Product Data Management – A Strategic Perspective*. 1st edn. Maj Engineering Publishing.

Krause, F.-L. *et al.* (1993) 'Product Modelling', *CIRP Annals - Manufacturing Technology*, 42(2), pp. 695–706. doi:10.1016/S0007-8506(07)62532-3.

Lagos, N. (2007) 'Knowledge-based product support systems', *PQDT - Global*, p. 288.

Langeveld, L. (2011) 'Product Design with Embodiment Design as a New Perspective', *Industrial Design - New Frontiers* [Preprint], (November 2011). doi:10.5772/20579.

Lawrence, W. (1989) 'Using Knowledge-Based Engineering', *Production*, p. 74.

Leu, M.C. (2016) 'NX10 FOR ENGINEERING DESIGN', *Design*, p. 207.

Lou, Z., Jiang, H. and Ruan, X. (2004) 'Development of an integrated knowledge-based system for mold-base design', *Journal of Materials Processing Technology*, 150(1–2), pp. 194–199. doi:10.1016/j.jmatprotec.2004.01.037.

Lovett, P.J., Ingram, A. and Bancroft, C.N. (2000) 'Knowledge-based engineering for SMEs - a methodology', *Journal of Materials Processing Technology*, 107(1–3), pp. 384–389. doi:10.1016/S0924-0136(00)00728-7.

- Männistö, T. *et al.* (1998) ‘Modelling generic product structures in STEP’, *CAD Computer Aided Design*, 30(14), pp. 1111–1118. doi:10.1016/S0010-4485(98)00067-0.
- Marjudi, S. *et al.* (2010) ‘A Review and Comparison of IGES and STEP’, *Proceedings Of World Academy Of Science, Engineering And Technology*, (January 2016), pp. 1013–1017.
- Martínez-Pellitero, S. *et al.* (2011) ‘A new process-based ontology for KBE system implementation: Application to inspection process planning’, *International Journal of Advanced Manufacturing Technology*, 57(1–4), pp. 325–339. doi:10.1007/s00170-011-3285-7.
- Martins, J.R.R.A. and Lambe, A.B. (2013) ‘Multidisciplinary Design Optimization: A Survey of Architectures’, *AIAA Journal*, 51(9), pp. 2049–2075. doi:10.2514/1.J051895.
- Mehmet Murat Baysal, Utpal Roy, Rachuri Sudarasan, Ram D. Sriram, K.W.L. (2005) ‘Product information exchange using Open assembly model: issues related to representation of geometric information’, in *Proceedings of The 2005 ASME International Mechanical Engineering Congress & Exposition*. Orlando.
- Melody Stokes (2001) *Managing Engineering Knowledge: MOKA - Methodology for Knowledge Based Engineering Applications*. Wiley-Blackwell.
- Memarsadeghi, N. and Varshney, A. (2020) ‘Virtual and Augmented Reality Applications in Science and Engineering’, *Computing in Science and Engineering*, 22(3), pp. 4–6. doi:10.1109/MCSE.2020.2987151.
- Michael J. Pratt (1988) ‘Synthesis of an optimal approach to form feature modelling’, in *Proceedings of the 1988 ASME International Computers in Engineering Conference and Exhibition*, pp. 263–274.
- Moorthy, A. and Vivekanand, S. (2007) ‘Integration of PLM with other concepts for empowering business environments’, in Garetti, M. *et al.* (eds) *Product Lifecycle Management (PLM’07) Assessing the industrial relevance*. Inderscience Enterprises Limited, pp. 93–106.
- NDIA Systems Engineering Division M&S Committee (2011) ‘Final Report of the Model Based Engineering (MBE) Subcommittee’, pp. 1–26.
- Object Management Group (2008) ‘OMG Systems Modeling Language (OMG SysML™)’.
- Object Management Group (2015) ‘OMG Unified Modeling Language’, *Informatik-Spektrum*

[Preprint]. doi:10.1007/s002870050092.

Okudan, G.E. and Medeiros, D.J. (2005) 'Facilitating collaborative design: A review on design representations and workstations', *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005*, 2 A, pp. 71–79. doi:10.1115/detc2005-85124.

Okudan, G.E. and Zappe, S.E. (2006) 'Teaching product design to non-engineers: A review of experience, opportunities and problems', *Technovation*, 26(11), pp. 1287–1293. doi:10.1016/j.technovation.2005.10.009.

Oldham, K. *et al.* (1998) 'MOKA-A Methodology and tools Oriented to Knowledge-based engineering Applications', *Proceedings of the Conference on Integration in Manufacturing. Göteborg, Sweden*, pp. 198–207.

Owen, J. (1997) *STEP: an introduction*. Information Geometers.

OWL Web Ontology Language Overview (2016). Available at: <http://www.w3.org/TR/owl-features/> (Accessed: 1 September 2016).

Pahl, G. *et al.* (2007) *Engineering design: a systematic approach*. doi:10.1007/978-1-84628-319-2.

Peltokoski, M., Lohtander, M. and Varis, J. (2015) 'The role of Product Data Management (PDM) in engineering design and the key differences between PDM and Product Lifecycle Management (PLM)', in *The 1st PDM forum for Finland-Russia collaboration*.

Peng, T.-K. and Trappey, A.J.C. (1998) 'A step toward STEP-compatible engineering data management: the data models of product structure and engineering changes', *Robotics and Computer-Integrated Manufacturing*, 14(2), pp. 89–109. doi:10.1016/S0736-5845(97)00016-1.

Reddy, E.J., Sridhar, C.N. V. and Rangadu, V.P. (2015) 'Knowledge Based Engineering: Notion, Approaches and Future Trends', *American Journal of Intelligent Systems*, 5(1), pp. 1–17. doi:10.5923/j.ajis.20150501.01.

Requicha, A.A.G. and Rossignac, J.R. (1992) 'Solid Modeling and Beyond', *IEEE Computer Graphics and Applications*, 12(5), pp. 31–44. doi:10.1109/38.156011.

Requicha, A.A.G. and Voelcker, H.B. (1982) 'Solid Modeling: A Historical Summary and Contemporary Assessment', *IEEE Computer Graphics and Applications*, 2(2), pp. 9–24.

doi:10.1109/MCG.1982.1674149.

Research and Markets (2020) *Global CAD Software Market (2020 to 2030) - by Technology, Model, Deployment, Level and Application*. Dublin.

Rocca, G. La (2011) *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*.

Rocca, G. La (2012) 'Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design', *Advanced Engineering Informatics*, 26(2), pp. 159–179. doi:10.1016/j.aei.2012.02.002.

La Rocca, G., Krakkers, L. and van Tooren, M.J.L. (2002) 'Development of an ICAD Generative Model for Blended Wing Body Aircraft Design', in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 4-6 September 2002, Atlanta, Georgia*, pp. 1–10. doi:10.2514/6.2002-5447.

La Rocca, G. and Tooren, M. van (2012) 'Knowledge based engineering to support complex product design', *Advanced Engineering Informatics*, 26(2), pp. 157–158. doi:10.1016/j.aei.2012.02.008.

Rocca, G. La and Tooren, M.J.L. Van (2007) 'Enabling distributed multidisciplinary design of complex products : a Knowledge Based Engineering approach', *International Journal for Design Research* [Preprint].

Rodrigues Da Silva, A. (2015) 'Model-driven engineering: A survey supported by the unified conceptual model', *Computer Languages, Systems and Structures*, 43, pp. 139–155. doi:10.1016/j.cl.2015.06.001.

Rosenfeld, L.W. (1995) 'Solid modeling and knowledge-based engineering', in *Handbook of solid modeling*. McGraw-Hill, Inc. New York, USA, pp. 91–911.

Ross, D.T. and Ward, J.E. (1968) *INVESTIGATIONS IN COMPUTER-AIDED DESIGN FOR NUMERICALLY CONTROLLED PRODUCTION*.

Roy, U., R. Sudarsan, R. D. Sriram, K. W. Lyons, M.R.D. (1999) 'Information Architecture for Design Tolerancing: From Conceptual to the Detail Design', in *Proceedings of the 1999 ASME Design Engineering Technical Conferences (25th Design Automation Conference)*.

Roy, R., Hinduja, S. and Teti, R. (2008) 'Recent advances in engineering design optimisation: Challenges and future trends', *CIRP Annals - Manufacturing Technology*,

57(2), pp. 697–715. doi:10.1016/j.cirp.2008.09.007.

Rumbaugh, J., Jacobson, I. and Booch, G. (1999) *The UML reference manual*, New York: Addison-Wesley.

Rynne, A. and Gaughran, W. (2007) ‘Cognitive modelling strategies for optimum design intent in parametric modelling (PM)’, *ASEE Annual Conference and Exposition, Conference Proceedings* [Preprint], (January 2007).

S. Rahimifard (1996) ‘A methodology to develop EXPRESS data models’, *International Journal of Computer Integrated Manufacturing*, 9(1), pp. 61–72.

S Shephard, M. *et al.* (2004) ‘Toward simulation-based design’, *Finite Elements in Analysis and Design*, 40(12), p. Pages 1575-1598. doi:10.1016/j.

Salomons, O.W., van Houten, F.J.A.M. and Kals, H.J.J. (1993) ‘Review of research in feature-based design’, *Journal of Manufacturing Systems*, 12(2), pp. 113–132. doi:10.1016/0278-6125(93)90012-I.

Salustri, F.A. (1996) ‘A formal theory for knowledge-based product model representation’, *Manufacturing Systems*, (519), pp. 1–19.

Salzman, H. (1989) ‘Computer-Aided Design: Limitations in Automating Design and Drafting’, 36(4), pp. 252–261.

Sanya, I.O. and Shehab, E.M. (2014) ‘An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry’, *International Journal of Production Research*, 53, pp. 1–27. doi:10.1080/00207543.2014.965352.

Schätzle, J. (2016) *Evaluate How the STEP Standard AP 242 Could Enable Knowledge Transfer between CAD and KBE Environments*. Norwegian University of Science and Technology.

Shapiro, V. (2002) ‘Solid Modeling’, *Handbook of computer aided geometric design*, 20, pp. 473–518. doi:DOI: 10.1016/B978-044451104-1/50021-6.

Shehab, E.M. and Abdalla, H.S. (2001) ‘Manufacturing cost modelling for concurrent product development’, *Robotics and Computer-Integrated Manufacturing*, 17(4), pp. 341–353. doi:10.1016/S0736-5845(01)00009-6.

Siemens PLM (2019a) ‘Siemens JT Format Reference v10.5’.

Siemens PLM (2019b) *STEP AP242 to NX geometry mapping*. Available at:

https://docs.plm.automation.siemens.com/tdoc/nx/11/nx_help#uid:xid1128422:index_xid458198:xid1182052:xid1182028 (Accessed: 1 January 2022).

Soyler, A. and Sala-Diakanda, S. (2010) 'A model-based systems engineering approach to capturing disaster management systems', *2010 IEEE International Systems Conference*, pp. 283–287. doi:10.1109/SYSTEMS.2010.5482340.

STEP AP203 and AP214 Protocols (2016). Available at: http://www.datakit.com/en/step_protocols.php (Accessed: 30 August 2016).

STEP AP242 Project (2014). Available at: <http://www.ap242.org/edition-2> (Accessed: 26 June 2020).

STEP AP242 protocol (2014). Available at: http://www.datakit.com/en/step_ap242.php (Accessed: 1 September 2016).

STEP Tools Incorporated (2008) *ST-Developer Tools Reference*. Available at: https://web.archive.org/web/20081109065451/http://www.steptools.com/support/stdev_docs/devtools/devtools-7.html (Accessed: 1 September 2019).

Stroud, I. (2006) *Boundary Representation Modelling Techniques*. 1st edn. Springer-Verlag London. doi:10.1007/978-1-84628-616-2.

Suresh, S. and Egbu, C. (2006) 'Key issues for implementing knowledge capture initiatives in small and medium enterprises in the UK construction industry', in *COBRA 2006 - Proceedings of the Annual Research Conference of the Royal Institution of Chartered Surveyors*.

Szykman, S., J.W.R. and R.D.S. (1999) 'THE REPRESENTATION OF FUNCTION IN COMPUTER-BASED DESIGN', in *Proceedings of the 1999 ASME Design Engineering Technical Conferences (11th International Conference on Design Theory and Methodology)*.

Tang, D. *et al.* (2001) 'STEP-based product modeling for concurrent stamped part and die development', *Computers in Industry*, 46(1), pp. 75–94. doi:10.1016/S0166-3615(01)00116-6.

Tay, F.E.H. and Gu, J. (2002) 'Product modeling for conceptual design support', *Computers in Industry*, 48, pp. 143–155. doi:10.1016/S0166-3615(02)00014-3.

Technosoft (2013) *Adaptive Modeling Language (AML)*, <Http://Www.Technosoft.Com/Aml.Php>. Available at: <http://www.technosoft.com/aml.php>

(Accessed: 27 August 2021).

Terpenney, J., Strong, S. and Wang, J. (2000) 'A METHODOLOGY FOR KNOWLEDGE DISCOVERY AND CLASSIFICATION', pp. 36–41.

Tilove, R.B. (1981) *Exploiting spatial and structural locality in geometric modelling*. University of Rochester.

Tolman, F.P. (1999) 'Product modeling standards for the building and construction industry: past, present and future', *Automation in Construction*, 8(3), pp. 227–235. doi:10.1016/S0926-5805(98)00073-9.

Tomiyama, T., Mäntylä, M. and Finger, S. (1995) *Knowledge Intensive CAD (Vol.1)*. CHAPMAN & HALL.

van Tooren, M. *et al.* (2005) 'Aircraft design support using knowledge engineering and optimisation techniques', *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 8, pp. 5074–5089. doi:10.2514/6.2005-2205.

Unity Technologies (2020) *Unity User Manual, Unity Documentation*. Available at: <https://docs.unity3d.com/Manual/index.html> (Accessed: 18 June 2020).

Unity Technologies (2021) *Unity solutions for architecture, engineering and construction*. Available at: <https://unity.com/solutions/architecture-engineering-construction>.

Velden, A. Van der (2007) 'CAD to CAE Process Automation Through iSIGHT-FD', in *Proceedings of the ASME Turbo Expo 2007*, p. 87093.

Walsh, V., Roy, R. and Bruce, M. (1988) 'Competitive by design', *Journal of Marketing Management*, 4(2), pp. 201–217.

Wang, F. *et al.* (2003) 'Towards modeling the evolution of product families', *ASME Computers and Information In Engineering Conference* [Preprint].

Wang, L. *et al.* (2002) 'Collaborative conceptual design - State of the art and future trends', *Computer-Aided Design*, 34(13), pp. 981–996.

Weisberg, D.E. (2008) *The Engineering Design Revolution - The People, Companies and Computer Systems That Changed Forever the Practice of Engineering*.

Wilson, D.J. (2006) 'How to Integrate Paper with CAD'. Open Archive white paper.

Wingård, L. (1991) *Introducing form features in product models: a step towards CAD/CAM*

with engineering terminology. PhD Dissertation, Computer System for Design and Development.

Wu, J.K., Liu, T.H. and Fischer, G.W. (1992) 'PDES/STEP-based information model for CAE and CAM integration', *International Journal of System Automation: Research & Application*, 2(4), pp. 375–394.

Xu, B. and Chen, N. (2009) 'An integrated method of CAD, CAE and multi-objective optimization', in *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*, pp. 1010–1014.

Yang, W.Z. *et al.* (2008) 'Recent development on product modelling: a review.', *International Journal of Production Research*, 46(21), pp. 6055–6085. doi:10.1080/00207540701343895.

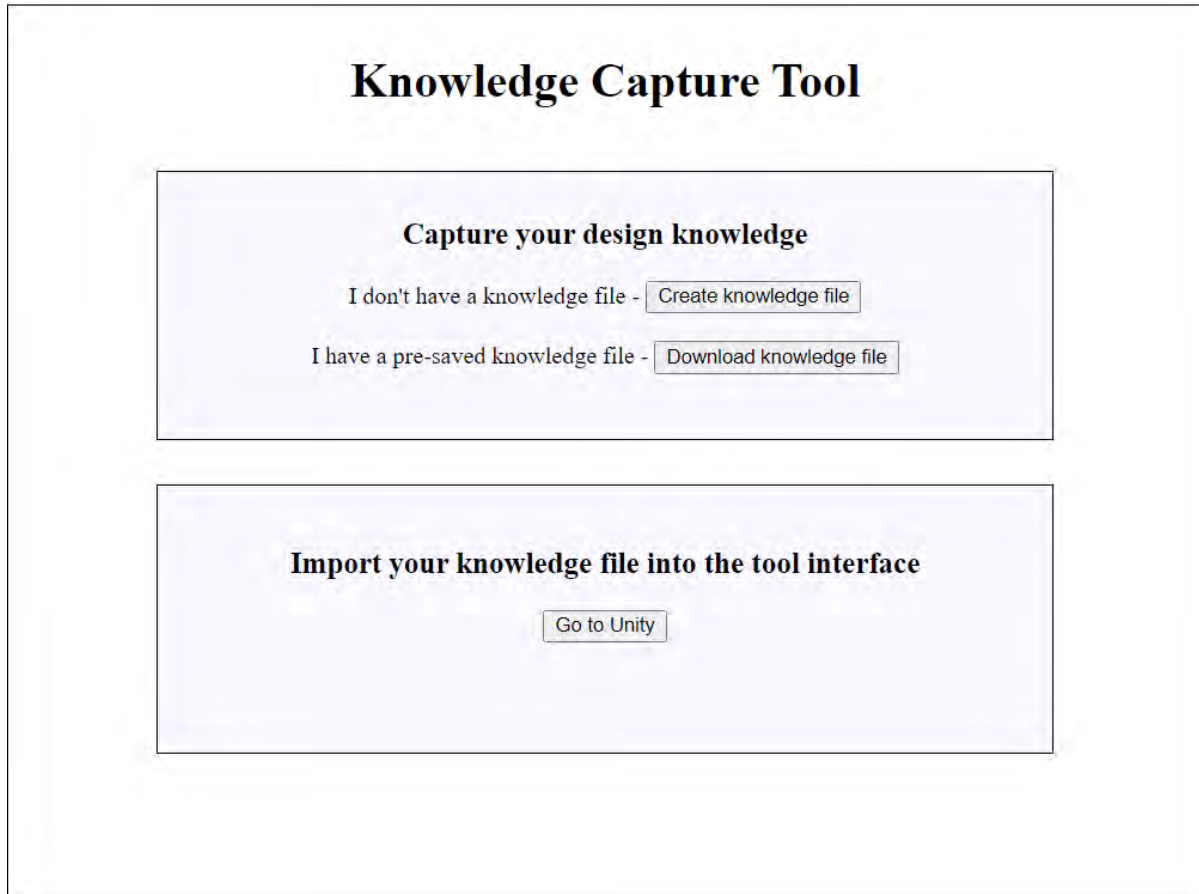
Yip-hoi, D.M. (2011) 'Teaching Surface Modeling to CAD/CAM Technologists', *Mechanical Engineering* [Preprint].

Yoshioka, M. (2001) 'Proposal of an Integrated Design Support Environment Based on', *Design engineering technical conference and Computers and Information in Engineering Conference* [Preprint].

Zha, X.F. and Du, H. (2002) 'A PDES/STEP-based model and system for concurrent integrated design and assembly planning', *CAD Computer Aided Design*, 34(14), pp. 1087–1110. doi:10.1016/S0010-4485(01)00186-5.

Appendix 1: Knowledge Capture Tool Interface Maximised View

a) Knowledge capture tool interface - function select



b) Knowledge capture tool interface – knowledge capture

Knowledge Capture Tool

How many parts are you designing? - Single Part

Start typing in the input field below:

Product Information	Design Intent :
Product ID: <input type="text"/> (if not known, a system id will be automatically created.)	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Product Name: <input type="text"/>	Name: <input type="text"/>
Product Description: <input type="text"/>	Description: <input type="text"/>
Product Type: <input type="text"/> (part or assembly)	
Function :	Form :
How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Name: <input type="text"/>	Name: <input type="text"/>
Description: <input type="text"/>	Description: <input type="text"/>
Property: <input type="text"/>	
Material:	Behaviour :
How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Name: <input type="text"/>	Name: <input type="text"/>
Description: <input type="text"/>	Description: <input type="text"/>
Property: <input type="text"/>	Property: <input type="text"/>
Rules :	Fit:
How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Name: <input type="text"/>	Description: <input type="text"/>
Description: <input type="text"/>	Property: <input type="text"/>
Equation: <input type="text"/>	
Formula: <input type="text"/>	
Property: <input type="text"/>	
Relationship:	Dimensions:
How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Relationship type: <input type="text"/>	Length: <input type="text"/> Unit: <input type="text"/>
Reference: <input type="text"/>	Width: <input type="text"/> Unit: <input type="text"/>
Description: <input type="text"/>	Height: <input type="text"/> Unit: <input type="text"/>
	Radius: <input type="text"/> Unit: <input type="text"/>
	Diameter: <input type="text"/> Unit: <input type="text"/>
Key Parameters	Constrains
How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>	How many would you like to input? <input type="text" value="1"/> <input type="button" value="v"/>
Name: <input type="text"/>	Name: <input type="text"/>
Value: <input type="text"/>	Description: <input type="text"/>
Unit: <input type="text"/>	Property: <input type="text"/>
	Constrain type: <input type="text"/> (internal or external)

Appendix 2: Scripts Used in Knowledge Mapping in Use Case 1

Platform: Unity, Programming language: C#

a) Knowledge mapping of block rules

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
public class BlockRules : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public Text length;
    public Text width;
    public Text height;

    public Text UpdatedL;
    public Text UpdatedW;
    public Text UpdatedH;

    public GameObject warning;
    public GameObject reason;

    public InputField Length;
    public InputField Width;
    public InputField Height;

    public Text warningtext;
    public Text reasontext;

    string rules01 = "Block Length L = Block Width W = Block Height H";

    public void checklength()
    {
        float l = float.Parse(Length.text.ToString());
        warningtext.text = "You have changed the length to " + l + "\n" + " The width and height
are also changed to "+l+ "\n" + "The change is affected by the following rule:";
        reasontext.text = KnowledgeFileStore.transfer.rules_k.ToString();

        UpdatedL.text = l.ToString();
        UpdatedW.text = l.ToString();
        UpdatedH.text = l.ToString();
    }
}
```

```

    }
    public void checkwidth()
    {
        float w = float.Parse(Width.text.ToString());
        warningtext.text = "You have changed the width to " + w + "\n" + " The length and height
are also changed to " + w + "\n" + "The change is affected by the following rule:";
        reasontext.text = KnowledgeFileStore.transfer.rules_k.ToString();

        UpdatedL.text = w.ToString();
        UpdatedW.text = w.ToString();
        UpdatedH.text = w.ToString();

    }
    public void checkheight()
    {

        float h = float.Parse(Height.text.ToString());
        warningtext.text = "You have changed the height to " + h + "\n" + " The length and width
are also changed to " + h + "\n" + "The change is affected by the following rule:";
        reasontext.text = KnowledgeFileStore.transfer.rules_k.ToString();

        UpdatedL.text = h.ToString();
        UpdatedW.text = h.ToString();
        UpdatedH.text = h.ToString();
    }

    // Update is called once per frame
    void Update()
    {

    }
}

```

b) Knowledge mapping of cylinder rules

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
public class CylinderRules : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public InputField Diameter;
    public InputField Height;
    public Text UpdatedDia;
    public Text UpdatedHeight;

    public GameObject warning;
    public GameObject reason;

    public Text warningtext;
    public Text reasontext;

    string rule01 = "Cylinder rule 01 : The height of cylinder should not be larger than 200.";
    string rule02 = "Cylinder rule 02 : The diameter of cylinder should not be larger than 80";

    public void checkheight()
    {

        float h = float.Parse(Height.text.ToString());

        if (h <= 200)
        {
            warningtext.text = "You can make this change. "+ "You have changed the height of
cylinder to " + h;
            reasontext.text = rule01;
            UpdatedHeight.text = h.ToString();
        }
        else
        {
            warningtext.text = "You cannot make this change.";
            reasontext.text = rule01;
        }
    }
}
```

```

public void checkdiameter()
{
    float d = float.Parse(Diameter.text.ToString());

    Debug.Log(Diameter.text.ToString());

    if (d <= 80)
    {
        warningtext.text = "You can make this change. " + "You have changed the diameter of
cylinder to " + d;
        reasontext.text = rule02;
        UpdatedDia.text = d.ToString();
    }
    else
    {
        warningtext.text = "You cannot make this change.";
        reasontext.text = rule02;
    }

}

// Update is called once per frame
void Update()
{

}
}

```

c) Knowledge mapping of cone rules

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
public class ConeRules : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public InputField Diameter;
    public InputField Height;

    public Text UpdatedDia;
    public Text UpdatedH;

    public GameObject warning;
    public GameObject reason;

    public Text warningtext;
    public Text reasontext;

    string rule01 = "Cone rule 01 : The base diameter of cone should be among 10,16,18,20 mm ";
    string rule02 = "Cone rule 02 : If the diameter of cone is less than 16 mm, the height should be
18 mm. If the diameter of cone is equal to or larger than 16 mm, the height should be 24 mm.";

    string[] diameterarray = { "10", "16", "18", "20" };

    public void checkdiameter()
    {

        float d = float.Parse(Diameter.text.ToString());
        string temp_d= d.ToString();

        foreach (string x in diameterarray){
            if (x.Equals(temp_d)){

                UpdatedDia.text = temp_d;
                if (d <= 16)
                {
                    warningtext.text = "You can make this change. " + "You have changed the diameter
to " + d + "\n" + "According to the rule 02, the height should be 18 mm";

                    reasontext.text = rule01 + rule02;
                    UpdatedH.text = "18";
                }
            }
        }
    }
}
```



```

    }
    else
    {
        warningtext.text = "You can make this change. " + "You have changed the diameter
to " + d + "\n" + "According to the rule 02, the height should be 24 mm";

        reasontext.text = rule01 + rule02;
        UpdatedH.text = "24";

    }
    break;

}
else
{

    warningtext.text = "You cannot make this change. According to the rule:";

    reasontext.text = rule01;

}
}
}

public void checkheight()
{
    warningtext.text = "The height of this cone is dominated by the diameter" + "\n" +
"According to the following rules:";

    reasontext.text = rule02;

}

// Update is called once per frame
void Update()
{

}
}
}

```

d) Knowledge mapping of sphere rules

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
public class SphereRules : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public InputField Diameter;
    public InputField Material;

    public Text UpdatedDia;
    public Text UpdatedM;

    public GameObject warning;
    public GameObject reason;

    public Text warningtext;
    public Text reasontext;

    string rule01 = "Sphere rule 01 : The diameter of sphere should be among
19,20,21,22,25,30,35,40 mm.";
    string rule02 = "Sphere rule 02 : The material of the steel ball should be among AIS 201, AIS
304, AIS 316 stainless steel.";

    string[] diameterarray = { "19", "20", "21", "22", "25", "30", "35", "40" };
    string[] materialarray = { "AIS 201", "AIS 304", "AIS 316" };

    public void checkmaterial()
    {

        string m = Material.text.ToString();

        foreach (string x in materialarray)
        {
            if (x.Equals(m))
            {
                warningtext.text = "You can make this change. " + "You have changed the material to "
+ m;

                reasontext.text = rule02;
                UpdatedM.text = m;
                break;
            }
        }
    }
}
```

```

else
{
    warningtext.text = "You cannot make this change. According to the rule:";

    reasontext.text = rule02;

}
}

}

public void checkdiameter()
{
    float d = float.Parse(Diameter.text.ToString());

    string temp_d = d.ToString();
    foreach (string x in diameterarray)
    {
        if (x.Equals(temp_d))
        {
            UpdatedDia.text = temp_d;
            warningtext.text = "You can make this change. " + "You have changed the diameter to "
+ d;

            reasontext.text = rule01;

            break;
        }

    }

    else
    {
        warningtext.text = "You cannot make this change. According to the rule:";

        reasontext.text = rule01;

    }
}

}

}

// Update is called once per frame
void Update()
{

}

}

```

Appendix 3: Scripts Used in Knowledge Mapping in Use Case 2

Platform: Unity, Programming language: C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
using System.Linq;
public class BoltRulesNew : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public string L;
    public string D1;
    public string b;
    public string k;
    public string s;
    public string e;

    public InputField BoltLength;
    public InputField ThreadSize;
    public InputField ThreadedShankLength;
    public InputField HeadDepth;

    public InputField WidthAcrossCorner;
    public InputField WidthAcrossFlats;

    public Text Updated_L;
    public Text Updated_D1;
    public Text Updated_b;
    public Text Updated_k;
    public Text Updated_s;
    public Text Updated_e;

    public bool check_k;
    public bool check_e;
    public bool check_s;
    public bool check_b;
    public bool check_L;

    public GameObject warning;
    public GameObject reason;
```

```
public Text warningtext;
public Text reasontext;
```

```
string rule01 = "When D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm;if L is larger than 200 mm, the thread length b should be 49mm.";
```

```
string rule02 = "When D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm.If L is less than 125 mm, the thread length b should be 34 mm;if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.";
```

```
string[] diameterarray = { "19", "20", "21", "22", "25", "30", "35", "40" };
string[] materialarray = { "AIS 201", "AIS 304", "AIS 316" };
```

```
string[] rule1_b = { "30", "36", "49" };
string[] rule2_b = { "34", "40", "53" };
string rule1_k = "7.5";
string rule1_e = "21.1";
string rule1_s = "19";
```

```
public void checkL()
{
```

```
    L = BoltLength.text.ToString();
```

```
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();
```

```
    string temp_D1 = ThreadSize.text.ToString();
```

```
    if (temp_D1 == "")
```

```
    {
```

```
        D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();
```

```
    }
```

```
    else
```

```
    {
```

```
        D1 = temp_D1;
```

```
    }
```

```
    float Value_L = float.Parse(L.ToString());
```

```
    Debug.Log(Value_L);
```

```
    Debug.Log(D1);
```

```
    if (D1 == "M12")
```

```
    {
```

```
        if (Value_L < 125)
```

```
        {
```

```
            b = "30";
```

```

        Updated_L.text = L;
        Updated_b.text = b;
        warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;

        reasontext.text = rule01;
        check_L = true;

    }

    if (Value_L > 200)
    {
        b = "49";
        Updated_L.text = L;
        Updated_b.text = b;
        warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
        reasontext.text = rule01;
        check_L = true;
    }

    if (Value_L <= 200 && Value_L >= 125)
    {
        b = "36";
        Updated_L.text = L;
        Updated_b.text = b;
        warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
        reasontext.text = rule01;
        check_L = true;
    }

}

}

public void checkb()
{
    b = KnowledgeFileStore.transfer.keyparap3_k.ToString();

    string temp_b = ThreadedShankLength.text.ToString();

    if (temp_b == "")
    {
        b = KnowledgeFileStore.transfer.keyparap3_k.ToString();
    }
    else
    {

```

```

    b = temp_b;
}

D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

string temp_D1 = ThreadSize.text.ToString();

if (temp_D1 == "")
{
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

}
else
{
    D1 = temp_D1;
}

if (D1 == "M12") {

    foreach (string x in rule1_b)
    {
        if (x.Equals(b))
        {
            check_b = true;

        }

    }

    if (check_b==true)
    {

        Debug.Log("CONTAINS B");

    }
    else
    {

        warningtext.text = "You cannot make this change. " + "b value conflict with rule01";

        reasontext.text = rule01;

        check_b = false;

    }

    if (b == "30")
    {
        warningtext.text = "You can make this change. " + "You have to change L to less than
125 mm";

```

```

        reasontext.text = rule01;

        Updated_b.text = "30";

        check_b = true;
    }

    if (b == "36")
    {
        warningtext.text = "You can make this change. " + "You have to change L to between
125 and 200 mm";

        reasontext.text = rule01;
        Updated_b.text = "36";
        check_b = true;
    }

    if (b == "49")
    {
        warningtext.text = "You can make this change. " + "You have to change L to larger than
200 mm";

        reasontext.text = rule01;

        Updated_b.text = "49";
        check_b = true;
    }

}

if (D1 == "M14")
{

    foreach (string x in rule2_b)
    {
        if (x.Equals(b))
        {
            check_b = true;

        }
    }

}

if (check_b == true)
{

    Debug.Log("CONTAINS B");
}

```



```

    }
else
{
    warningtext.text = "You cannot make this change. " + "b value conflict with rule02";

    reasontext.text = rule02;

    check_b = false;

}

if (b == "34")
{
    warningtext.text = "You can make this change. " + "You have to change L to less than
125 mm";

    reasontext.text = rule02;

    Updated_b.text = "34";

    check_b = true;

}

if (b == "40")
{
    warningtext.text = "You can make this change. " + "You have to change L to between
125 and 200 mm";

    reasontext.text = rule02;
    Updated_b.text = "40";
    check_b = true;

}

if (b == "53")
{
    warningtext.text = "You can make this change. " + "You have to change L to larger than
200 mm";

    reasontext.text = rule02;

    Updated_b.text = "53";
    check_b = true;

}

}

}

```

```

public void checkk()
{
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    string temp_D1 = ThreadSize.text.ToString();

    if (temp_D1 == "")
    {
        D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    }
    else
    {
        D1 = temp_D1;
    }

    k = KnowledgeFileStore.transfer.keyparap4_k.ToString();

    string temp_k = HeadDepth.text.ToString();

    if (temp_k == "")
    {
        k = KnowledgeFileStore.transfer.keyparap4_k.ToString();

    }
    else
    {
        k = temp_k;
    }

    if (D1 == "M12")
    {
        if (k == "7.5")
        {
            warningtext.text = "You can make this change. ";

            reasontext.text = rule01;

            Updated_k.text = "7.5";

        }
        else
        {

```

```

        warningtext.text = "You cannot make this change. " + "k value conflict with rule01";

        reasontext.text = rule01;

    }

}
if (D1 == "M14")
{
    if (k == "8.8")
    {

        warningtext.text = "You can make this change. ";

        reasontext.text = rule02;

        Updated_k.text = "8.8";

    }
    else
    {

        warningtext.text = "You cannot make this change. " + "k value conflict with rule02";

        reasontext.text = rule02;

    }

}

}

}

public void checke()
{

    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    string temp_D1 = ThreadSize.text.ToString();

    if (temp_D1 == "")
    {
        D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    }
    else
    {
        D1 = temp_D1;
    }
}

```

```

}

e = KnowledgeFileStore.transfer.keyparap5_k.ToString();
string temp_e = WidthAcrossCorner.text.ToString();

if (temp_e == "")
{
    e = KnowledgeFileStore.transfer.keyparap5_k.ToString();

}
else
{

    e = temp_e;

}

if (D1 == "M12")
{
    if (e == "21.1")
    {

        warningtext.text = "You can make this change. ";

        reasontext.text = rule01;
        Updated_e.text = "21.1";

    }
    else
    {

        warningtext.text = "You cannot make this change. " + "e value conflict with rule01";

        reasontext.text = rule01;

    }

}

if (D1 == "M14")
{
    if (e == "24.49")
    {

        warningtext.text = "You can make this change. ";

        reasontext.text = rule02;
        Updated_e.text = "24.49";

    }
    else

```

```

    {
        warningtext.text = "You cannot make this change. " + "e value conflict with rule02";
        reasontext.text = rule02;
    }

}

}

public void checks()
{
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    string temp_D1 = ThreadSize.text.ToString();

    if (temp_D1 == "")
    {
        D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

    }
    else
    {
        D1 = temp_D1;
    }

    s = KnowledgeFileStore.transfer.keyparap6_k.ToString();

    string temp_s = WidthAcrossFlats.text.ToString();

    if (temp_s == "")
    {
        s = KnowledgeFileStore.transfer.keyparap6_k.ToString();

    }
    else
    {

        s = temp_s;
    }

    if (D1 == "M12")
    {

```

```

if (s == "19")
{
    warningtext.text = "You can make this change. ";
    reasontext.text = rule01;
    Updated_s.text = "19";
}
else
{
    warningtext.text = "You cannot make this change. " + "s value conflict with rule01";
    reasontext.text = rule01;
}
}

if (D1 == "M14")
{
    if (s == "22")
    {
        warningtext.text = "You can make this change. ";
        reasontext.text = rule02;
        Updated_s.text = "22";
    }
    else
    {
        warningtext.text = "You cannot make this change. " + "s value conflict with rule02";
        reasontext.text = rule02;
    }
}

}

public void checkD1()
{
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();
    string temp_D1 = ThreadSize.text.ToString();
}

```

```

if (temp_D1 == "")
{
    D1 = KnowledgeFileStore.transfer.keyparap2_k.ToString();
}
else
{
    D1 = temp_D1;
}

k = KnowledgeFileStore.transfer.keyparap4_k.ToString();

string temp_k = HeadDepth.text.ToString();

if (temp_k == "")
{
    k = KnowledgeFileStore.transfer.keyparap4_k.ToString();
}
else
{
    k = temp_k;
}

e = KnowledgeFileStore.transfer.keyparap5_k.ToString();
string temp_e = WidthAcrossCorner.text.ToString();

if (temp_e == "")
{
    e = KnowledgeFileStore.transfer.keyparap5_k.ToString();
}
else
{
    e = temp_e;
}

s = KnowledgeFileStore.transfer.keyparap6_k.ToString();

string temp_s = WidthAcrossFlats.text.ToString();

if (temp_s == "")
{
    s = KnowledgeFileStore.transfer.keyparap6_k.ToString();
}

```

```

}
else
{
    s = temp_s;
}

L = KnowledgeFileStore.transfer.keyparap1_k.ToString();

string temp_L = BoltLength.text.ToString();

if (temp_L == "")
{
    L = KnowledgeFileStore.transfer.keyparap1_k.ToString();

}
else
{
    L = temp_L;
}

float Value_L = float.Parse(L.ToString());

b = KnowledgeFileStore.transfer.keyparap3_k.ToString();

string temp_b = ThreadedShankLength.text.ToString();

if (temp_b == "")
{
    b = KnowledgeFileStore.transfer.keyparap3_k.ToString();

}
else
{
    b = temp_b;
}

if (D1 == "M12")
{

```



```

if (k == "7.5")
{
    check_k = true;
}
else
{
    warningtext.text = "You cannot make this change. " + "k value conflict with rule01";
    reasontext.text = rule01;

    check_k = false;
}
if ( e== "21.1")
{
    check_e = true;

}
else
{
    check_e = false;
    warningtext.text = "You cannot make this change. " + "e value conflict with rule01";
    reasontext.text = rule01;

    check_k = false;
}
if (s=="19")
{
    check_s = true;

}
else
{
    warningtext.text = "You cannot make this change. " + "s value conflict with rule01";
    reasontext.text = rule01;

    check_s = false;
}

```

```

if (check_k&&check_e&&check_s&&check_b&&check_b == true)
{
    warningtext.text = "You can make this change. ";
    reasontext.text = rule01;

}
else
{

    warningtext.text = "You cannot make this change. ";
    reasontext.text = rule01;

}

if (Value_L < 125)
{
    b = "30";
    Updated_L.text = L;
    Updated_b.text = b;
    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;

    reasontext.text = rule01;
    check_L = true;

}

if (Value_L > 200)
{
    b = "49";
    Updated_L.text = L;
    Updated_b.text = b;
    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
    reasontext.text = rule01;
    check_L = true;

}

if (Value_L <= 200 && Value_L >= 125)
{
    b = "36";
    Updated_L.text = L;
    Updated_b.text = b;
    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
    reasontext.text = rule01;
    check_L = true;

}

}

if (D1 == "M14")
{

```

```

if (Value_L < 125)
{
    b = "34";

    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;

    reasontext.text = rule02;
    check_L = true;

}

if (Value_L > 200)
{
    b = "53";

    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
    reasontext.text = rule02;
    check_L = true;
}

if (Value_L <= 200 && Value_L >= 125)
{
    b = "40";

    warningtext.text = "You can make this change. " + "You have changed L to " + L + "
.You have to change the b to " + b;
    reasontext.text = rule02;
    check_L = true;
}

foreach (string x in rule2_b)
{
    if (x.Equals(b))
    {
        check_b = true;

    }
    else
    {

    }
}

if (check_b == true)
{

    Debug.Log("CONTAINS B");
}

```

```

}
else
{

    warningtext.text = "You cannot make this change. " + "b value conflict with rule02";

    reasontext.text = rule01;

    check_b = false;

}

if (k == "8.8")
{

    //warningtext.text = "You can make this change. ";

    //reasontext.text = rule01;

    //Updated_k.text = "7.5";
    check_k = true;

}
else
{

    warningtext.text = "You cannot make this change. " + "k value conflict with rule01";

    reasontext.text = rule02;

    check_k = false;

}
if (e == "24.49")
{
    check_e = true;

}
else
{

    warningtext.text = "You cannot make this change. " + "e value conflict with rule01";

    reasontext.text = rule02;

    check_e = false;

}
if (s == "22")
{

```

```

        check_s = true;

    }
    else
    {

        warningtext.text = "You cannot make this change. " + "s value conflict with rule01";

        reasontext.text = rule02;

        check_s = false;

    }

    if (check_k && check_e && check_s && check_b == true)
    {
        warningtext.text = "You can make this change. ";
        reasontext.text = rule02;

        Updated_D1.text = "M14";

    }
    else
    {

        warningtext.text = "You cannot make this change. ";
        reasontext.text = rule02;

        Debug.Log(k+e+s+b+L);

    }

}

}

// Update is called once per frame
void Update()
{

}

}

```

Appendix 4: Scripts Used in Knowledge Mapping in Use Case 3

Platform: Unity, Programming language: C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;
using System.Linq;
public class WheelAssemblyRulesNew : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public string D1;
    public string D2;
    public string D3;
    public string D4;
    public string L1;
    public string L2;
    public string L3;
    public string L4;

    public string S1;
    public string S2;
    public string S3;
    public string S4;

    public string L;
    public string S;

    public InputField Input_D1;
    public InputField Input_D2;
    public InputField Input_D3;
    public InputField Input_D4;

    public InputField Input_L1;
    public InputField Input_L2;
    public InputField Input_L3;
    public InputField Input_L4;

    public InputField Input_S1;
    public InputField Input_S2;
    public InputField Input_S3;
    public InputField Input_S4;

    public InputField Input_L;
```

```

public InputField Input_S;

public Text Updated_D1;
public Text Updated_D2;
public Text Updated_D3;
public Text Updated_D4;

public Text Updated_L1;
public Text Updated_L2;
public Text Updated_L3;
public Text Updated_L4;

public Text Updated_S1;
public Text Updated_S2;
public Text Updated_S3;
public Text Updated_S4;

public Text Updated_L;
public Text Updated_S;

public GameObject warning;
public GameObject reason;

public Text warningtext;
public Text reasontext;

string Assembly_rule01 = "D1 = D3, when D1 is changed, D3 needs to be changed as well, and vice versa.";
string Assembly_rule02 = "D2 = D4, when D2 is changed, D4 needs to be changed as well, and vice versa.";
string Assembly_rule03 = "L1 = S1, when L1 is changed, S1 needs to be changed as well, and vice versa.";
string Assembly_rule04 = "L2 = S2, when L2 is changed, S2 needs to be changed as well, and vice versa.";
string Assembly_rule05 = "L3 = S3, when L3 is changed, S3 needs to be changed as well, and vice versa.";
string Assembly_rule06 = "L4 = S4, when L4 is changed, S4 needs to be changed as well, and vice versa.";
string Assembly_rule07 = "L = S, when L is changed, S needs to be changed as well, and vice versa.";

string Wheel_rule01 = "L1 = L4, when L1 is changed, L4 needs to be changed as well, and vice versa.";
string Wheel_rule02 = "L2 = L3, when L2 is changed, L3 needs to be changed as well, and vice versa.";

string Tire_rule01 = "S1 = S4, when S1 is changed, S4 needs to be changed as well, and vice versa.";

```

```
string Tire_rule02 = "S2 = S3, when S2 is changed, S3 needs to be changed as well, and vice versa.";
```

```
string warningY = "You can make this change.";  
string warningN = "Warning! You cannot make this change!";
```

```
string[] diameterarray = { "19", "20", "21", "22", "25", "30", "35", "40" };  
string[] materialarray = { "AIS 201", "AIS 304", "AIS 316" };
```

```
string[] rule1_b = { "30", "36", "49" };  
string[] rule2_b = { "34", "40", "53" };  
string rule1_k = "7.5";  
string rule1_e = "21.1";  
string rule1_s = "19";
```

```
public void checkD1D3()  
{
```

```
    D1 = KnowledgeFileStore.transfer.keyparap1_k.ToString();  
    string temp_D1 = Input_D1.text.ToString();
```

```
    if (temp_D1 == "")
```

```
    {  
        D1 = KnowledgeFileStore.transfer.keyparap1_k.ToString();
```

```
    }  
    else
```

```
    {  
        D1 = temp_D1;
```

```
    }
```

```
    D3 = KnowledgeFileStore.transfer.keyparap3_k.ToString();  
    string temp_D3 = Input_D3.text.ToString();
```

```
    if (temp_D3 == "")
```

```
    {  
        D3 = KnowledgeFileStore.transfer.keyparap3_k.ToString();
```

```
    }  
    else
```

```
    {  
        D3 = temp_D3;
```

```
    }
```

```
    if (D1 == D3)
```

```
    {  
        Updated_D1.text = D1;  
        Updated_D3.text = D3;  
        warningtext.text = warningY;
```



```

reason.SetActive(true);
reasontext.text = "Allowed by rules:" + Assembly_rule01;

}
else
{

warningtext.text = warningN;
reason.SetActive(true);
reasontext.text = Assembly_rule01;

}

}

public void checkD2D4()
{

D2 = KnowledgeFileStore.transfer.keyparap2_k.ToString();
string temp_D2 = Input_D2.text.ToString();

if (temp_D2 == "")
{
D2 = KnowledgeFileStore.transfer.keyparap2_k.ToString();

}
else
{
D2 = temp_D2;

}

D4 = KnowledgeFileStore.transfer.keyparap4_k.ToString();
string temp_D4 = Input_D4.text.ToString();

if (temp_D4 == "")
{
D4 = KnowledgeFileStore.transfer.keyparap4_k.ToString();

}
else
{
D4 = temp_D4;

}

if (D2 == D4)
{

Updated_D2.text = D2;
Updated_D4.text = D4;

```

```

warningtext.text = warningY;
reason.SetActive(true);
reasontext.text = "Allowed by rules:" + Assembly_rule02;

}
else
{
warningtext.text = warningN;
reason.SetActive(true);
reasontext.text = Assembly_rule02;

}

}

public void checkL1S1()
{

L1 = KnowledgeFileStore.transfer.keyparap5_k.ToString();
string temp_L1 = Input_L1.text.ToString();

if (temp_L1 == "")
{
L1 = KnowledgeFileStore.transfer.keyparap5_k.ToString();

}
else
{
L1 = temp_L1;

}

S1 = KnowledgeFileStore.transfer.keyparap9_k.ToString();
string temp_S1 = Input_S1.text.ToString();

if (temp_S1 == "")
{
S1 = KnowledgeFileStore.transfer.keyparap9_k.ToString();

}
else
{
S1 = temp_S1;

}

L4 = KnowledgeFileStore.transfer.keyparap8_k.ToString();
string temp_L4 = Input_L4.text.ToString();

```

```

if (temp_L4 == "")
{
    L4 = KnowledgeFileStore.transfer.keyparap8_k.ToString();

}
else
{
    L4 = temp_L4;
}

S4 = KnowledgeFileStore.transfer.keyparap12_k.ToString();
string temp_S4 = Input_S4.text.ToString();

if (temp_S4 == "")
{
    S4 = KnowledgeFileStore.transfer.keyparap12_k.ToString();

}
else
{
    S4 = temp_S4;
}

string temp_reason = "" + "\n";
bool temp_warning_condition1;
bool temp_warning_condition2;
bool temp_warning_condition3;

if (L1 == S1)
{
    reason.SetActive(true);

    temp_warning_condition1 = false;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);

    temp_reason = temp_reason + Assembly_rule03;

    temp_warning_condition1 = true;
}

```

```

if (L1 == L4)
{
    reason.SetActive(true);

    temp_warning_condition2 = false;
}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);

    temp_reason = temp_reason + Wheel_rule01;

    temp_warning_condition2 = true;
}

if (S4 == S1)
{
    reason.SetActive(true);

    temp_warning_condition3 = false;
}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);

    temp_reason = temp_reason + Tire_rule01;

    temp_warning_condition3 = true;
}

if(temp_warning_condition1 == true || temp_warning_condition2 == true ||
temp_warning_condition3 == true)
{
    warningtext.text = warningN;
    reasontext.text = temp_reason;
}
else
{
    Updated_L1.text = L1;
    Updated_S1.text = S1;
    Updated_S4.text = S4;
    Updated_L4.text = L4;
}

```

```

        warningtext.text = warningY;
        reasontext.text = "Allowed by rules:" + Assembly_rule03+ Wheel_rule01+ Tire_rule01;
    }
}

public void checkL2S2()
{

    L2 = KnowledgeFileStore.transfer.keyparap6_k.ToString();
    string temp_L2 = Input_L2.text.ToString();

    if (temp_L2 == "")
    {
        L2 = KnowledgeFileStore.transfer.keyparap6_k.ToString();

    }
    else
    {
        L2 = temp_L2;
    }

    S2 = KnowledgeFileStore.transfer.keyparap10_k.ToString();
    string temp_S2 = Input_S2.text.ToString();

    if (temp_S2 == "")
    {
        S2 = KnowledgeFileStore.transfer.keyparap10_k.ToString();

    }
    else
    {
        S2 = temp_S2;
    }

    L3 = KnowledgeFileStore.transfer.keyparap7_k.ToString();
    string temp_L3 = Input_L3.text.ToString();

    if (temp_L3 == "")
    {
        L3 = KnowledgeFileStore.transfer.keyparap7_k.ToString();

    }
    else
    {
        L3 = temp_L3;
    }
}

```

```

S3 = KnowledgeFileStore.transfer.keyparap11_k.ToString();
string temp_S3 = Input_S3.text.ToString();

if (temp_S3 == "")
{
    S3 = KnowledgeFileStore.transfer.keyparap11_k.ToString();

}
else
{
    S3 = temp_S3;
}

string temp_reason = "" + "\n";
bool temp_warning_condition1;
bool temp_warning_condition2;
bool temp_warning_condition3;
bool temp_warning_condition4;

if (L2 == S2)
{
    //warningtext.text = warningY;
    reason.SetActive(true);
    // reasontext.text = Assembly_rule04;
    temp_warning_condition1 = false;
}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);
    // reasontext.text = Assembly_rule04;

    temp_reason = temp_reason + Assembly_rule04;

    temp_warning_condition1 = true;
}

if (L2 == L3)
{

    reason.SetActive(true);

    temp_warning_condition2 = false;

}
else
{
    warningtext.text = warningN;

```

```

reason.SetActive(true);

temp_reason = temp_reason + Wheel_rule02;
temp_warning_condition2 = true;
}

if (S2 == S3)
{
    reason.SetActive(true);

    temp_warning_condition3 = false;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);
    // reasontext.text = Tire_rule02;

    temp_reason = temp_reason + Tire_rule02;
    temp_warning_condition3 = true;

}

if (S3 == L3)
{
    reason.SetActive(true);

    temp_warning_condition4 = false;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);

    temp_reason = temp_reason + Assembly_rule05;
    temp_warning_condition4 = true;

}

if (temp_warning_condition1 == true || temp_warning_condition2 == true ||
temp_warning_condition3 == true || temp_warning_condition4 == true)

```

```

    {
        warningtext.text = warningN;
        reasontext.text = temp_reason;
    }
    else
    {

        Updated_S2.text = S2;
        Updated_S3.text = S3;
        Updated_L3.text = L3;
        Updated_L2.text = L2;
        warningtext.text = warningY;
        reasontext.text = "Allowed by rules:" + Assembly_rule04 + Wheel_rule02 + Tire_rule02
+ Assembly_rule05;
    }
}

```

```

public void checkL3S3()
{

    L3 = KnowledgeFileStore.transfer.keyparap7_k.ToString();
    string temp_L3 = Input_L3.text.ToString();

    if (temp_L3 == "")
    {
        L3 = KnowledgeFileStore.transfer.keyparap7_k.ToString();

    }
    else
    {
        L3 = temp_L3;

    }

    S3 = KnowledgeFileStore.transfer.keyparap11_k.ToString();
    string temp_S3 = Input_S3.text.ToString();

    if (temp_S3 == "")
    {
        S3 = KnowledgeFileStore.transfer.keyparap11_k.ToString();

    }
    else
    {
        S3 = temp_S3;

    }

    if (L3 == S3)
    {

```



```

warningtext.text = warningY;
reason.SetActive(true);
reasontext.text = Assembly_rule05;

}
else
{
warningtext.text = warningN;
reason.SetActive(true);
reasontext.text = Assembly_rule05;

}

L2 = KnowledgeFileStore.transfer.keyparap6_k.ToString();
string temp_L2 = Input_L2.text.ToString();

if (temp_L2 == "")
{
L2 = KnowledgeFileStore.transfer.keyparap6_k.ToString();

}
else
{
L2 = temp_L2;

}

S2 = KnowledgeFileStore.transfer.keyparap10_k.ToString();
string temp_S2 = Input_S2.text.ToString();

if (temp_S2 == "")
{
S2 = KnowledgeFileStore.transfer.keyparap10_k.ToString();

}
else
{
S2 = temp_S2;

}

if (L2 == L3)
{
warningtext.text = warningY;
reason.SetActive(true);
reasontext.text = Wheel_rule02;

```

```

    }
    else
    {
        warningtext.text = warningN;
        reason.SetActive(true);
        reasontext.text = Wheel_rule02;

    }

    if (S2 == S3)
    {
        warningtext.text = warningY;
        reason.SetActive(true);
        reasontext.text = Tire_rule02;

    }
    else
    {
        warningtext.text = warningN;
        reason.SetActive(true);
        reasontext.text = Tire_rule02;

    }
}

public void checkL4S4()
{
    L4 = KnowledgeFileStore.transfer.keyparap8_k.ToString();
    string temp_L4 = Input_L4.text.ToString();

    if (temp_L4 == "")
    {
        L4 = KnowledgeFileStore.transfer.keyparap8_k.ToString();

    }
    else
    {
        L4 = temp_L4;

    }

    S4 = KnowledgeFileStore.transfer.keyparap12_k.ToString();
    string temp_S4 = Input_S4.text.ToString();

    if (temp_S4 == "")
    {
        S4 = KnowledgeFileStore.transfer.keyparap12_k.ToString();
    }
}

```

```

}
else
{
    S4 = temp_S4;
}

if (L4 == S4)
{
    warningtext.text = warningY;
    reason.SetActive(true);
    reasontext.text = Assembly_rule06;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);
    reasontext.text = Assembly_rule06;

}

}

L1 = KnowledgeFileStore.transfer.keyparap5_k.ToString();
string temp_L1 = Input_L1.text.ToString();

if (temp_L1 == "")
{
    L1 = KnowledgeFileStore.transfer.keyparap5_k.ToString();

}
else
{
    L1 = temp_L1;

}

S1 = KnowledgeFileStore.transfer.keyparap9_k.ToString();
string temp_S1 = Input_S1.text.ToString();

if (temp_S1 == "")
{
    S1 = KnowledgeFileStore.transfer.keyparap9_k.ToString();

}
else
{
    S1 = temp_S1;

}
}

```

```

if (L1 == L4)
{
    warningtext.text = warningY;
    reason.SetActive(true);
    reasontext.text = Wheel_rule01;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);
    reasontext.text = Wheel_rule01;

}

}

if (S1 == S4)
{
    warningtext.text = warningY;
    reason.SetActive(true);
    reasontext.text = Tire_rule01;

}
else
{
    warningtext.text = warningN;
    reason.SetActive(true);
    reasontext.text = Tire_rule01;

}

}

}

public void checkLS()
{

    L = KnowledgeFileStore.transfer.keyparap13_k.ToString();
    string temp_L = Input_L.text.ToString();

    if (temp_L == "")
    {
        L = KnowledgeFileStore.transfer.keyparap13_k.ToString();

    }
    else

```

```

    {
        L = temp_L;
    }

    S = KnowledgeFileStore.transfer.keyparap14_k.ToString();
    string temp_S = Input_S.text.ToString();

    if (temp_S == "")
    {
        S = KnowledgeFileStore.transfer.keyparap14_k.ToString();

    }
    else
    {
        S = temp_S;
    }

    if (L == S)
    {
        warningtext.text = warningY;
        reason.SetActive(true);
        reasontext.text = Assembly_rule07;

    }
    else
    {
        warningtext.text = warningN;
        reason.SetActive(true);
        reasontext.text = Assembly_rule07;

    }

}
// Update is called once per frame
void Update()
{
}
}

```

Appendix 5: Scripts Used in Parsing Data From Knowledge File

Platform: Unity, Programming language: C#

a) Storing data from knowledge file

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class KnowledgeFileStore : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    public static KnowledgeFileStore transfer;

    public string id_k = "";
    public string name_k = "";
    public string description_k = "";
    public string product_type_k = "";
    public string design_intention_k = "";
    public string function_k = "";
    public string form_k = "";
    public string behaviour_k = "";
    public string fit_k = "";
    public string relation_k = "";
    public string material_k = "";
    public string rules_k = "";
    public string dimension_k = "";
    public string length_k = "";
    public string width_k = "";
    public string height_k = "";
    public string diameter_k = "";
    public string radius_k = "";
    public string keyparameter_k = "";

    public List<string> keyparalist;

    public string keyparap1_k = "";
    public string keyparap2_k = "";
    public string keyparap3_k = "";
    public string keyparap4_k = "";
    public string keyparap5_k = "";
    public string keyparap6_k = "";
    public string keyparap7_k = "";
    public string keyparap8_k = "";
    public string keyparap9_k = "";
    public string keyparap10_k = "";
    public string keyparap11_k = "";
```

```
public string keyparap12_k = "";  
public string keyparap13_k = "";  
public string keyparap14_k = "";
```

```
public string keyparap1v_k = "";  
public string keyparap2v_k = "";  
public string keyparap3v_k = "";  
public string keyparap4v_k = "";  
public string keyparap5v_k = "";  
public string keyparap6v_k = "";  
public string keyparap7v_k = "";  
public string keyparap8v_k = "";  
public string keyparap9v_k = "";  
public string keyparap10v_k = "";  
public string keyparap11v_k = "";  
public string keyparap12v_k = "";  
public string keyparap13v_k = "";  
public string keyparap14v_k = "";
```

```
public string keyparap1v_o = "";  
public string keyparap2v_o = "";  
public string keyparap3v_o = "";  
public string keyparap4v_o = "";  
public string keyparap5v_o = "";  
public string keyparap6v_o = "";  
public string keyparap7v_o = "";  
public string keyparap8v_o = "";  
public string keyparap9v_o = "";  
public string keyparap10v_o = "";  
public string keyparap11v_o = "";  
public string keyparap12v_o = "";  
public string keyparap13v_o = "";  
public string keyparap14v_o = "";
```

```
public string keypara_userinput1 = "";  
public string keypara_userinput2 = "";  
public string keypara_userinput3 = "";  
public string keypara_userinput4 = "";  
public string keypara_userinput5 = "";  
public string keypara_userinput6 = "";  
public string keypara_userinput7 = "";  
public string keypara_userinput8 = "";
```

```
public string keypara_userinput9 = "";  
public string keypara_userinput10 = "";  
public string keypara_userinput11 = "";  
public string keypara_userinput12 = "";  
public string keypara_userinput13 = "";  
public string keypara_userinput14 = "";
```

```
public string formula_k = "";  
public string equation_k = "";
```

```
// Update is called once per frame

void Awake()
{
    if (transfer == null)
    {
        DontDestroyOnLoad(gameObject);
        transfer = this;
    }
    else if (transfer != this)
    {
        Destroy(gameObject);
    }
}
void Update()
{
}
}
```


b) Parsing data for visualisation

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Xml;
using UnityEngine.UI;

public class XMLparse_WheelAssembly : MonoBehaviour
{

    public TextAsset xmlRawFile; // Knowledge File
    public Text RuleNumberText;
    public Text uiText2;

    string productid = "";
    string productname = "";
    string productdescription = "";
    string producttype = "";

    string total_designintention = "";
    string total_designintention2 = "";
    string total_rules = "";
    string total_rules2 = "";
    string total_function = "";
    string total_function2 = "";
    string total_form = "";
    string total_form2 = "";
    string total_behaviour = "";
    string total_behaviour2 = "";
    string total_material = "";
    string total_material2 = "";
    string total_fit = "";
    string total_fit2 = "";
    string total_relation = "";
    string total_relation2 = "";

    string total_dimension = "";
    string total_dimension2 = "";

    string total_keypara = "";
    string total_keypara2 = "";

    string id_tok = ""; // id stored in knowledge file
    string name_tok = "";
    string description_tok = "";
    string product_type_tok = "";
    string designintention_tok = "";
    string rules_tok = "";
```

```

string function_tok = "";
string form_tok = "";
string behaviour_tok = "";
string fit_tok = "";
string relation_tok = "";
string material_tok = "";
string dimension_tok = "";
string keypara_tok = "";
//This variables are for Knowledge tab
//public Text IDtextHere;

// dimensions***** //

string p_length = "";
string p_width = "";
string p_height = "";
string p_radius = "";
string p_diameter = "";
string p_material = "";

string p_keypara_1 = "";
string p_keypara_2 = "";
string p_keypara_3 = "";
string p_keypara_4 = "";
string p_keypara_5 = "";
string p_keypara_6 = "";

string p_keypara_7 = "";
string p_keypara_8 = "";
string p_keypara_9 = "";
string p_keypara_10 = "";
string p_keypara_11 = "";
string p_keypara_12 = "";

string p_keypara_13 = "";
string p_keypara_14 = "";

// Interface*****//
public InputField IDPropertyField;
public InputField NamePropertyField;
public InputField DescriptionPropertyField;
public Text ProductTypeField;

public InputField DesignIntentionPropertyField;
public InputField FunctionPropertyField;
public InputField FormPropertyField;
public InputField FitPropertyField;
public InputField MaterialPropertyField;
public InputField BehaviourPropertyField;
public InputField RulesPropertyField;
public Text RulesNumber;
public InputField RelationPropertyField;

public InputField DimensionPropertyField;
public InputField LengthPropertyField;

```

```

public InputField WidthPropertyField;
public InputField HeightPropertyField;
public InputField DiameterPropertyField;
public InputField KeyparaPropertyField;
public InputField FormulaPropertyField;
public InputField EquationPropertyField;

// Interface - Change //
public InputField Length_oPropertyField;
public InputField Width_oPropertyField;
public InputField Height_oPropertyField;
public InputField Diameter_oPropertyField;
public InputField Radius_oPropertyField;

public InputField Keypara_op1PropertyField;
public InputField Keypara_op2PropertyField;
public InputField Keypara_op3PropertyField;
public InputField Keypara_op4PropertyField;
public InputField Keypara_op5PropertyField;
public InputField Keypara_op6PropertyField;
public InputField Keypara_op7PropertyField;
public InputField Keypara_op8PropertyField;
public InputField Keypara_op9PropertyField;
public InputField Keypara_op10PropertyField;
public InputField Keypara_op11PropertyField;
public InputField Keypara_op12PropertyField;
public InputField Keypara_op13PropertyField;
public InputField Keypara_op14PropertyField;

public InputField Keypara_userinput1_PropertyField;
public InputField Keypara_userinput2_PropertyField;
public InputField Keypara_userinput3_PropertyField;
public InputField Keypara_userinput4_PropertyField;
public InputField Keypara_userinput5_PropertyField;
public InputField Keypara_userinput6_PropertyField;
public InputField Keypara_userinput7_PropertyField;
public InputField Keypara_userinput8_PropertyField;

public InputField Keypara_userinput9_PropertyField;
public InputField Keypara_userinput10_PropertyField;
public InputField Keypara_userinput11_PropertyField;
public InputField Keypara_userinput12_PropertyField;
public InputField Keypara_userinput13_PropertyField;
public InputField Keypara_userinput14_PropertyField;

public InputField Material_oPropertyField;

// Use this for initialization
void Start()
{

```

```

total_designintention = "";
total_rules2 = "";
total_function2 = "";
total_form2 = "";
total_behaviour2 = "";
total_fit2 = "";
total_relation2 = "";
total_material2 = "";
total_dimension2 = "";
total_keypara2 = "";
}

```

```

public void parseXml_File(string xmlData)

```

```

{
total_designintention = "";
total_rules2 = "";
total_function2 = "";
total_form2 = "";
total_behaviour2 = "";
total_fit2 = "";
total_relation2 = "";
total_material2 = "";
total_dimension2 = "";
total_keypara2 = "";

XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load(new StringReader(xmlData));
// xmlTag for searching *****//
string xmlTag = "//knowledge"; //this is the search tags
string xmlTag_product = "//knowledge/product";
string xmlTag_designintention = "knowledge/product/design_intention";
string xmlTag_rules = "//knowledge/product/rules"; // search product tag 可以用
string xmlTag_function = "//knowledge/product/function_"; // search product tag 可以用
string xmlTag_form = "//knowledge/product/form";
string xmlTag_behaviour = "//knowledge/product/behaviour";
string xmlTag_material = "//knowledge/product/material";
string xmlTag_fit = "//knowledge/product/fit";
string xmlTag_relation = "//knowledge/product/relationship";
string xmlTag_dimension = "//knowledge/product/dimension";
string xmlTag_keypara = "//knowledge/product/dimension/keyparameter";

// XmlNodeList *****//

XmlNodeList rulesnodes = xmlDoc.SelectNodes(xmlTag_rules);
XmlNodeList productnodes = xmlDoc.SelectNodes(xmlTag_product);
XmlNodeList designintentionnodes = xmlDoc.SelectNodes(xmlTag_designintention);
XmlNodeList functionnodes = xmlDoc.SelectNodes(xmlTag_function);
XmlNodeList formnodes = xmlDoc.SelectNodes(xmlTag_form);

```

```

XmlNodeList behaviournodes = xmlDoc.SelectNodes(xmlTag_behaviour);
XmlNodeList materialnodes = xmlDoc.SelectNodes(xmlTag_material);
XmlNodeList fitnodes = xmlDoc.SelectNodes(xmlTag_fit);
XmlNodeList relationnodes = xmlDoc.SelectNodes(xmlTag_relation);
XmlNodeList dimensionnodes = xmlDoc.SelectNodes(xmlTag_dimension);
XmlNodeList keyparanodes = xmlDoc.SelectNodes(xmlTag_keypara);
// product id name description and type
*****//
foreach (XmlNode node in productnodes)
{
    XmlNode product_id = node["id"];
    XmlNode product_n = node["name"];// 可以用
    XmlNode product_d = node["description"];

    productid = product_id.InnerXml.ToString();
    productname = product_n.InnerXml.ToString();
    productdescription = product_d.InnerXml.ToString();

}

producttype = xmlDoc.SelectSingleNode(xmlTag_product).Attributes["type"].Value;
ProductTypeField.text = producttype.ToString();
IDPropertyField.text = productid;
NamePropertyField.text = productname;
DescriptionPropertyField.text = productdescription;

KnowledgeFileStore.transfer.id_k = productid;
KnowledgeFileStore.transfer.name_k = productname;
KnowledgeFileStore.transfer.description_k = productdescription;
KnowledgeFileStore.transfer.product_type_k = producttype;
Debug.Log(producttype + productid + productname + productdescription);

// rules*****//
foreach (XmlNode node in rulesnodes)
{
    XmlNode rulesn = node["name"];
    XmlNode rulesd = node["description"];
    XmlNode rulesf = node["formula"];
    XmlNode rulese = node["equation"];

    total_rules = "Name:" + rulesn.InnerXml.ToString() + "\nDescription:" +
rulesd.InnerXml.ToString() + "\nFormula:" + rulesf.InnerXml.ToString() + "\nEquation:" +
rulese.InnerXml.ToString() + "\n";

    //Debug.Log(total_rules);
    total_rules2 = total_rules2 + total_rules;
    total_rules = "";
}

```

```

    rules_tok = total_rules2.ToString();
}

RulesPropertyField.text = rules_tok;
KnowledgeFileStore.transfer.rules_k = rules_tok;
Debug.Log(rules_tok);

// Count the rules number*****

int rules_nodeCount = rulesnodes.Count;
RulesNumber.text = rules_nodeCount.ToString();
Debug.Log("rules number count: " + rules_nodeCount);
//

// design intent*****//

foreach (XmlNode node in designintentionnodes)
{
    XmlNode designintention_n = node["name"];
    XmlNode designintention_d = node["description"];

    total_designintention = "Name:" + designintention_n.InnerXml.ToString() +
"\nDescription:" + designintention_d.InnerXml.ToString() + "\n";

    //Debug.Log(total_rules);
    total_designintention2 = total_designintention + total_designintention2;
    total_designintention = "";

    designintention_tok = total_designintention2.ToString();
}

DesignIntentionPropertyField.text = designintention_tok;
KnowledgeFileStore.transfer.design_intention_k = designintention_tok;
Debug.Log(designintention_tok);

// function*****//

foreach (XmlNode node in functionnodes)
{
    XmlNode function_n = node["name"];
    XmlNode function_d = node["description"];
    XmlNode function_p = node["property"];

    total_function = "Name:" + function_n.InnerXml.ToString() + "\nDescription:" +
function_d.InnerXml.ToString() + "\nProperty:" + function_p.InnerXml.ToString() + "\n";
}

```

```

//Debug.Log(total_rules);
total_function2 = total_function + total_function2;
total_function = "";

function_tok = total_function2.ToString();

}

FunctionPropertyField.text = function_tok;
KnowledgeFileStore.transfer.function_k = function_tok;
Debug.Log(function_tok);

// form*****//

foreach (XmlNode node in formnodes)
{

XmlNode form_n = node["name"];
XmlNode form_d = node["description"];

total_form = "Name:" + form_n.InnerXml.ToString() + "\nDescription:" +
form_d.InnerXml.ToString() + "\n";

total_form2 = total_form2 + total_form;
total_form = "";

form_tok = total_form2.ToString();

}

FormPropertyField.text = form_tok;
KnowledgeFileStore.transfer.form_k = form_tok;
Debug.Log(form_tok);

// behaviour *****//

foreach (XmlNode node in behaviournodes)
{

XmlNode behaviour_n = node["name"];
XmlNode behaviour_d = node["description"];
XmlNode behaviour_p = node["property"];

total_behaviour = "Name:" + behaviour_n.InnerXml.ToString() + "\nDescription:" +
behaviour_d.InnerXml.ToString() + "\nProperty:" + behaviour_p.InnerXml.ToString() + "\n";

total_behaviour2 = total_behaviour2 + total_behaviour;
total_behaviour = "";

behaviour_tok = total_behaviour2.ToString();

}

```

```

BehaviourPropertyField.text = behaviour_tok;
KnowledgeFileStore.transfer.behaviour_k = behaviour_tok;
Debug.Log(behaviour_tok);

// fit *****//

foreach (XmlNode node in fitnodes)
{
    XmlNode fit_n = node["name"];
    XmlNode fit_d = node["description"];
    XmlNode fit_p = node["property"];

    total_fit = "Name:" + fit_n.InnerXml.ToString() + "\nDescription:" +
fit_d.InnerXml.ToString() + "\nProperty:" + fit_p.InnerXml.ToString() + "\n";

    total_fit2 = total_fit2 + total_fit;
    total_fit = "";

    fit_tok = total_fit2.ToString();
}

FitPropertyField.text = fit_tok;
KnowledgeFileStore.transfer.fit_k = fit_tok;
Debug.Log(fit_tok);

// relationship *****//

foreach (XmlNode node in relationnodes)
{
    XmlNode relation_p = node["parent"];
    XmlNode relation_c = node["children"];
    XmlNode relation_r = node["reference"];
    XmlNode relation_d = node["description"];

    total_relation = "Parent:" + relation_p.InnerXml.ToString() + "\nChildren:" +
relation_c.InnerXml.ToString() + "\nReference:" + relation_r.InnerXml.ToString() +
"\nDescription:" + relation_d.InnerXml.ToString() + "\n";

    total_relation2 = total_relation2 + total_relation;
    total_relation = "";

    relation_tok = total_relation2.ToString();
}

RelationPropertyField.text = relation_tok;
KnowledgeFileStore.transfer.relation_k = relation_tok;
Debug.Log(relation_tok);

// material *****//

```



```

foreach (XmlNode node in materialnodes)
{
    XmlNode material_n = node["name"];
    XmlNode material_d = node["description"];
    XmlNode material_p = node["property"];

    total_material = "Name:" + material_n.InnerXml.ToString() + "\nDescription:" +
material_d.InnerXml.ToString() + "\nProperty:" + material_p.InnerXml.ToString() + "\n";

    total_material2 = total_material2 + total_material;
    total_material = "";

    material_tok = total_material2.ToString();
    p_material = material_n.InnerXml.ToString();
}

MaterialPropertyField.text = material_tok;
Material_oPropertyField.text = p_material;
KnowledgeFileStore.transfer.material_k = material_tok;
Debug.Log(material_tok);

foreach (XmlNode node in dimensionnodes)
{
    XmlNode unit = node["unit"];
    XmlNode length = node["length"];
    XmlNode width = node["width"];
    XmlNode height = node["height"];
    XmlNode radius = node["radius"];
    XmlNode diameter = node["diameter"];
    XmlNode keypara = node["keyparameter"];

    total_dimension = "Unit:" + unit.InnerXml.ToString() + "\nlength:" +
length.InnerXml.ToString() + "\nwidth:" + width.InnerXml.ToString() + "\nheight:" +
height.InnerXml.ToString() + "\nradius:" + radius.InnerXml.ToString() + "\ndiameter:" +
diameter.InnerXml.ToString() + "\nkey para:" + keypara.InnerXml.ToString() + "\n";

    //Debug.Log(total_rules);
    total_dimension2 = total_dimension2 + total_dimension;
    total_dimension = "";

    dimension_tok = total_dimension2.ToString();

    p_length = length.InnerXml.ToString();
    p_width = width.InnerXml.ToString();
    p_height = height.InnerXml.ToString();
    p_radius = radius.InnerXml.ToString();
    p_diameter = diameter.InnerXml.ToString();
}

```

```

KnowledgeFileStore.transfer.length_k = length.InnerXml.ToString();
KnowledgeFileStore.transfer.width_k = width.InnerXml.ToString();
KnowledgeFileStore.transfer.height_k = height.InnerXml.ToString();
KnowledgeFileStore.transfer.diameter_k = diameter.InnerXml.ToString();

}

Length_oPropertyField.text = p_length;
Width_oPropertyField.text = p_width;
Height_oPropertyField.text = p_height;
Diameter_oPropertyField.text = p_diameter;

DimensionPropertyField.text = dimension_tok;
KnowledgeFileStore.transfer.dimension_k = dimension_tok;
Debug.Log(dimension_tok);

// keypara *****//
foreach (XmlNode node in keyparanodes)
{
    XmlNode keypara_name = node["name"];
    XmlNode keypara_value = node["value"];

    string value = keypara_value.InnerXml.ToString();
    total_keypara = "Name:" + keypara_name.InnerXml.ToString() + "\nValue:" +
keypara_value.InnerXml.ToString() + "\n";

    //Debug.Log(total_rules);
    total_keypara2 = total_keypara + total_keypara;
    total_keypara = "";

    keypara_tok = total_keypara2.ToString();

    KnowledgeFileStore.transfer.keyparalist.Add(value);

}

KeyparaPropertyField.text = keypara_tok;
KnowledgeFileStore.transfer.keyparameter_k = keypara_tok;

Keypara_op1PropertyField.text = KnowledgeFileStore.transfer.keyparalist[0];
Keypara_op2PropertyField.text = KnowledgeFileStore.transfer.keyparalist[1];
Keypara_op3PropertyField.text = KnowledgeFileStore.transfer.keyparalist[2];
Keypara_op4PropertyField.text = KnowledgeFileStore.transfer.keyparalist[3];
Keypara_op5PropertyField.text = KnowledgeFileStore.transfer.keyparalist[4];
Keypara_op6PropertyField.text = KnowledgeFileStore.transfer.keyparalist[5];

Keypara_op7PropertyField.text = KnowledgeFileStore.transfer.keyparalist[6];
Keypara_op8PropertyField.text = KnowledgeFileStore.transfer.keyparalist[7];
Keypara_op9PropertyField.text = KnowledgeFileStore.transfer.keyparalist[8];

```

```
Keypara_op10PropertyField.text = KnowledgeFileStore.transfer.keyparalist[9];
Keypara_op11PropertyField.text = KnowledgeFileStore.transfer.keyparalist[10];
Keypara_op12PropertyField.text = KnowledgeFileStore.transfer.keyparalist[11];
Keypara_op13PropertyField.text = KnowledgeFileStore.transfer.keyparalist[12];
Keypara_op14PropertyField.text = KnowledgeFileStore.transfer.keyparalist[13];
```

```
KnowledgeFileStore.transfer.keyparap1_k = Keypara_op1PropertyField.text;
KnowledgeFileStore.transfer.keyparap2_k = Keypara_op2PropertyField.text;
KnowledgeFileStore.transfer.keyparap3_k = Keypara_op3PropertyField.text;
KnowledgeFileStore.transfer.keyparap4_k = Keypara_op4PropertyField.text;
KnowledgeFileStore.transfer.keyparap5_k = Keypara_op5PropertyField.text;
KnowledgeFileStore.transfer.keyparap6_k = Keypara_op6PropertyField.text;
KnowledgeFileStore.transfer.keyparap7_k = Keypara_op7PropertyField.text;
```

```
KnowledgeFileStore.transfer.keyparap8_k = Keypara_op8PropertyField.text;
KnowledgeFileStore.transfer.keyparap9_k = Keypara_op9PropertyField.text;
KnowledgeFileStore.transfer.keyparap10_k = Keypara_op10PropertyField.text;
KnowledgeFileStore.transfer.keyparap11_k = Keypara_op11PropertyField.text;
KnowledgeFileStore.transfer.keyparap12_k = Keypara_op12PropertyField.text;
KnowledgeFileStore.transfer.keyparap13_k = Keypara_op13PropertyField.text;
KnowledgeFileStore.transfer.keyparap14_k = Keypara_op14PropertyField.text;
```

```
Debug.Log(keypara_tok);
```

```
}
```

```
public void ReadKnowledge()
```

```
{
```

```
    string data = xmlRawFile.text;
    parseXml_File(data);
```

```
}
```

```
}
```

A Game-Based Product Modelling Environment for Non-Engineer

Guolong Zhong, Venkatesh Chennam Vijay, Ilias Oraifige

Abstract—In the last 20 years, Knowledge Based Engineering (KBE) has shown its advantages in product development in different engineering areas such as automation, mechanical, civil and aerospace engineering in terms of digital design automation and cost reduction by automating repetitive design tasks through capturing, integrating, utilising and reusing the existing knowledge required in various aspects of the product design. However, in primary design stages, the descriptive information of a product is discrete and unorganized while knowledge is in various forms instead of pure data. Thus, it is crucial to have an integrated product model which can represent the entire product information and its associated knowledge at the beginning of the product design. One of the shortcomings of the existing product models is a lack of required knowledge representation in various aspects of product design and its mapping to an interoperable schema. To overcome the limitation of the existing product model and methodologies, two key factors are considered. First, the product model must have well-defined classes that can represent the entire product information and its associated knowledge. Second, the product model needs to be represented in an interoperable schema to ensure a steady data exchange between different product modelling platforms and CAD software. This paper introduced a method to provide a general product model as a generative representation of a product, which consists of the geometry information and non-geometry information, through a product modelling framework. The proposed method for capturing the knowledge from the designers through a knowledge file provides a simple and efficient way of collecting and transferring knowledge. Further, the knowledge schema provides a clear view and format on the data that needed to be gathered in order to achieve a unified knowledge exchange between different platforms. This study used a

game-based platform to make product modelling environment accessible for non-engineers. Further the paper goes on to test use case based on the proposed game-based product modelling environment to validate the effectiveness among non-engineers.

Keywords—Game-based learning, knowledge based engineering, product modelling, design automation.

I. INTRODUCTION

THE rapid development of technology has generated higher demands of industrial development capacity, productivity and agile response to market. They all drive the industry to design and produce more complex products at lower cost and with less time. Product design is widely regarded as one of the most important steps in the development of a product. Without a design, there would be no product. The United Kingdom Department of Trade and Industry (DTI) identified that investing money and resources at the design stage yields the biggest return on investment of a product [1]. Product design automation is an impactful differentiator among business competitors as a reduction in time and manpower cost of a product design will allow companies to make flexible business strategies and be price competitive in the market. Companies and industries are able to achieve more sales and high revenues by accelerating the formulation of the new product and offering multiple product variants and options to customers, in the way of reusing and modifying an existing product. However, due to incomplete representation of products and components at the detail design stage, most of the design tasks can only be performed by experienced designers [2]. Modification of a complex CAD model of an engineering product is usually difficult and time-consuming for a new designer. For a non-engineer without enough knowledge of this product,

Mr. Guolong Zhong is a PhD student with the Birmingham City University, Millennium Point, Birmingham, B4 7XG, United Kingdom (e-mail: guolong.zhong@bcu.ac.uk).

Dr. Venkatesh Vijay is the lead of Knowledge-based Engineering Lab, Birmingham City University, Millennium Point, Birmingham, B4 7XG, United Kingdom (e-mail: venkatesh.vijay@bcu.ac.uk).

Prof. Ilias Oraifige is the Head of Centre of Engineering, Birmingham City University, Millennium Point, Birmingham, B4 7XG, United Kingdom (e-mail: ilias.oraifige@bcu.ac.uk).

it is not easy to completely understand the propose of each design detail in the existing CAD model or to unravel the complex design references created by other designers before making changes or adding additional geometric features. Thus, at the beginning of the product design, an integrated product model which can represent the entire product information and its associated knowledge will enable designers to work on the design tasks without previous design experience. Time and cost on tutorial sessions and trainings for new designers could be saved. Also, a product model that captures all required design knowledge can perform as a knowledge base for different elements, varying from originality to final product. In this manner, even a complex product, which has features and structures that are special and cannot be modelled or physically produced in straightforward way, can be represented by a comprehensive model.

In primary design stages, the descriptive information of a product is discrete and unorganized while knowledge is in various forms instead of pure data. Capture, store and share the knowledge of experienced designers through a generic product model turns-out to be a challenge. This research focuses on developing a product modelling method that captures and maps the design knowledge into a framework for enhancing the automation of the engineering product design. The proposed framework presents a knowledge capture method in capturing design engineers' knowledge with respect to geometric and non-geometric information. Further explains how the captured knowledge are modelled and mapped into a product modelling framework for enhancing the product design in a knowledge- based environment.

II. BACKGROUND

A product model is an information representation that provides data contributing to build the form (geometry and topology), function (intent) and behaviour (load resistance, etc.) of a product in a modelling process [3]. It is employed throughout the entire lifecycle of a product to structure product data and design information. To date, engineering in product design uses four main approaches to generate a product model, which are: (i) solid

product modelling, (ii) feature-based product modelling, (iii) knowledge-based product modelling and (iv) integrated product modelling [4]. In this paper, integrated product modelling is regarded as part of knowledge- based modelling methodology as it is a functional combination of different modelling methods.

A. Solid Product Modelling

Solid product modelling [5] is a technique that uses mathematical principles and computer modelling to achieve precise representation of three-dimensional objects . It is now a mature tool that is widely implemented in product modelling field [4]. There are two common methods of solid product modelling [9]: boundary representation (B-rep) [6] and constructive solid geometry (CSG) [5]. In B-rep method, the product is divided into a number of faces bounded by edges. In turn, the edges are bounded by two vertices at last. The B-rep method provides a fast display of a product geometry with basic information about the faces, edges and vertices [6]. The CSG method breaks the product into a binary tree of basic solids for example, cylinders, spheres, cones and cubes etc. The product itself in CSG is considered as a combination of those basic solids by utilizing union, difference and intersection operations [4]. Therefore, it can be seen that both B-rep and CSG present a clear and simple data structure of a product. However, [7] pointed out that the weakness of solid product modelling lies in providing all necessary information for an entire product development lifecycle. Solid product modelling works in a different way from a human product designer because it can only create models with basic geometric information such as dimension, tolerance etc. For a complete product lifecycle, more necessary information is still required, for example, "how the product will be manufactured", "what the function of the product is", etc.

B. Feature-Based Modelling

Feature-based product modelling [7] is seen as a well- developed extension of solid product modelling . The "Feature" here is defined by [8] as information sets that refer to aspects of form or other attributes of a part, such that these sets can be used in reasoning about design, performance or manufacture

of the part or assemblies they constitute. Chen and Wei [7] mentioned that feature-based product modelling shows great advantages over conventional solid modelling methods, for example, capturing design intents, relating functionality with product geometry and working on high level shapes instead of geometric details, etc. In the past twenty years, much research has been conducted by using features to support product design [9], [10] and assembling process [11]. However, a product from feature-based modelling is not able to transfer knowledge such as expertise and experience to other designers. A product modelling methodology that could capture and reuse the involved knowledge is in demand.

C. Knowledge-Based Product Modelling

The methodology that allows to capture and structure knowledge about a product design and its design process is known as knowledge-based engineering or KBE. Knowledge-based product modelling is characterised by capturing and reusing engineering knowledge such as human expertise, product and process knowledge in modelling process.

In 1989, Lawrence [12] suggested a public recognition of the potential of knowledge-based engineering - "Although it's not yet widely known, knowledge-based engineering is having a profound effect on how a few companies are speeding their products to market". In 1990, Jurit et al. [13] integrated frame-based and rule-based knowledge representation in a feature-based modelling system. It is considered as an early attempt to combine knowledge and expertise into a product modelling system. In the last 20 years, KBE has shown its advantages in product development in different engineering areas such as automation, civil engineering and aerospace engineering in terms of modelling and cost saving [14], [15].

The use of KBE methods and techniques has played an important role in design automation for the development of a product in industry [16], [17]. However, the current design and implementation of the KBE systems are usually platform specific and domain dependent. These kinds of KBE systems are usually developed in a particular programming language (i.e. C, C++, Java) which means the principle design parameters, rules, constraints are all

defined and written in code. Thus, a change of a platform will always require an entire re-write of the KBE system. Furthermore, transparency of KBE systems is also necessary in order to avoid black-box problems that happen in the communication between different KBE applications [18], [19]. This requires a framework that can provide adaptable, structured and reusable knowledge bases.

D. STEP Standard

To improve interoperability of the product model among CAD software and to avoid the data conversion faults, such as data missing and mismatching, an international standard needs to be followed. STEP is the international standard for exchange of product data. It addresses product data from mechanical and electrical design, geometric dimensions and tolerances, analysis and manufacturing, as well as additional information specific to various industries such as automotive, aerospace, building construction, ship, oil and gas, process plants and others. Because of the complexity, STEP is divided into smaller component parts, including a series of 'Application Protocols' or APs, each covering a particular industrial domain. And each APs is titled by the domain that it applies to [20]. The research of Yang et al. [4] shows the potential of using STEP as an interoperable data standard to exchange and share product data in different engineering environment. Although the readability of the STEP data model is low due to the tedious STEP definition, modelling product with STEP can provide a standardized, neutral data exchange between CAD systems.

III. CHALLENGES

In a traditional product modelling environment, the designer is able to create and provide the visible geometry information in the product model. However, only geometry information is not enough to describe a product model completely. The utilization of the existing associated knowledge of a product can greatly reduce the unnecessary re-analysis, re-design, and re-planning, simplify the modelling tasks, and ensure the modelling quality. Having said that, capturing and transferring the knowledge of experienced designers are difficult. According to [21], the main challenge that exists in

small and medium companies and industries for implementing knowledge capture initiatives is lack of awareness of knowledge capture benefits. Because of this, individuals, small and medium enterprises are lack of vision and strategy as well as structure for knowledge capture. They have strong reliance on informal networks and collaboration to locate the repository of knowledge. Therefore, one important issue of knowledge-based product modelling is to classify, structure and manage the captured knowledge. Since there is no clear formalised link between a generic product model and an interoperable format in KBE environment, it is essential to provide well-defined knowledge classes and a formalized knowledge capture method for individuals, enterprises and industries to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats. In this paper, geometric data contained in a CAD file are regarded as “geometry information” and non-geometry information, such as experience, expertise and design rules, is called “knowledge”.

This research aims to provide a knowledge-based product modelling framework and method that could capture and integrate the geometry information and the associated knowledge applied by designers, into a generative product model. To achieve this target, two key factors are considered. Firstly, the generated product model must have well-defined classes that can represent the entire product information and its associated knowledge. Then, the captured knowledge in the product model needs to be represented in an interoperable schema to ensure a steady data exchange between different product modelling platforms and CAD software. The rest of the paper will discuss in detail on how the knowledge from the design engineer is captured and mapped into the framework for automation.

IV. A KNOWLEDGE-BASED PRODUCT MODELLING FRAMEWORK

A. Conceptual Framework

A game-based product modelling environment does not mean turning the product modelling process into games. It means applying KBE methodology in a product modelling environment with the help of a

game-based platform. To develop such a product modelling environment, a knowledge-based product modelling framework (Fig. 1) has been proposed in this research.

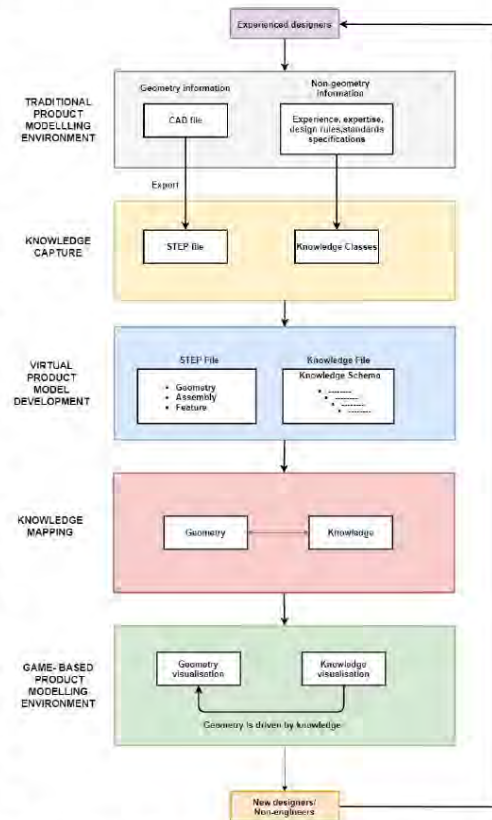


Fig. 1 Knowledge-based product modelling framework

This framework consists of four major blocks: knowledge capture, virtual product model development, knowledge mapping and product visualization and validation. The knowledge capture block provides an environment where the non-geometry knowledge that applied by experienced designers in the product modelling process is captured. The geometry information of the product can be captured by exporting the CAD model into an interoperable format - STEP.

After performing knowledge capture, a knowledge

file that contains well decomposed non-geometric classes and properties is generated, and a geometry file with the geometric data that describes the product's geometry is also acquired. A virtual product model will be formed by utilising the knowledge that captured in the previous stage. This virtual product model is a comprehensive representation of the product with all its essential non-geometric and geometric information. Knowledge Mapping is the most significant stage of the framework, which builds the connection and interaction between the non-geometric information and geometry. In this stage, non-geometric information is transferred from the knowledge file to object-oriented programming environment, where key parameters and design rules are linked to the product geometry. The data from the knowledge capture tool are exported into a knowledge file in an XML format and parsed using the object-oriented programming. The last stage is the visualization and validation stage, where the developed virtual product model is visualised and checked for the correct representation of the initial model. Testing use cases is performed for identifying and evaluating the effectiveness of the proposed framework. If design engineer changes one dimension of the geometry of the initial product model, the virtual product model checks the rules that determine this geometry and propagates the rules and changes to the design engineer. In this case, design engineer can check what will be affected if geometry is to be changed in this model and the constraints of these changes.

B. Virtual Product Model

Variance of product models always results in extra work in transferring data from different models. Therefore, in order to make the representation as robust as possible without having to predefine attributes that might be relevant only in a given domain, it is necessary to create a generic and platform independent product model that can be used in all product design stages and different platforms.

In this research, a Virtual Product Model (VPM) is developed as a generative representation of a product to enhance the knowledge representation of the product model but maintain generic. The meta class sets (Fig. 1) defined for the VPM are derived from literature review analysis, previous related research

in product modelling [22]-[29], own experience of product modelling in industry and expert's knowledge. Fig. 2 shows the how a product is represented with knowledge classes in VPM.



Fig. 1 VPM meta knowledge classes

When the VPM is transferred between different platforms, the geometry information will be stored and transmitted through the STEP file and the knowledge will be stored and transmitted through an XML document. To ensure the data structure of the XML document, an XML schema has been developed based on the framework of VPM. This XML document is named as Knowledge File (KF) in this research and will be used as an independent format for data exchange.

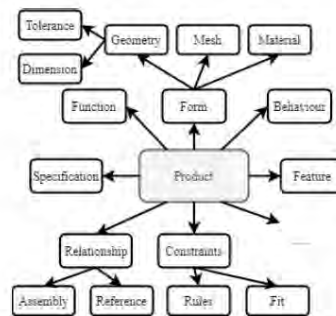


Fig. 2 A structure view of the product represented by the VPM atomic blocks

Knowledge Capture Tool

How many parts are you designing?
 [Back to Home Page](#)

Start typing a name in the input field below:
 Name:
 Suggestions: Bolt

Start typing a function in the input field below:
 Function:
 Suggestions: Fasten

Start typing a form in the input field below:
 Form:
 Suggestions: Cylinder

Start typing a material in the input field below:
 Material:
 Suggestions: Steel

Start typing a behaviour in the input field below:
 Behaviour:
 Your input: The bolt head locks the bolt in the place and nut is applied at the end.

Start typing rules in the input field below:
 Rule:
 Your input: If bolt Length is 25mm, the hole depth should be 36mm.

(a)

```

knowledge category="part"
name="Bolt" /name
function="Fasten" /function
form="Cylinder, external male thread" /form
material="Steel" /material
behaviour="The bolt head locks the bolt in the place and nut is applied at the end" /behaviour
rules="If bolt Length is 25mm, the hole depth should be 36mm" /rules
/knowledge
    
```

(b)

Fig. 3 (a) Knowledge capture tool interface (b) knowledge file generated from the knowledge capture tool

The developed framework was applied into a game-based product modelling prototype system which was developed in Unity 3D software. Unity 3D has a license-free version for personal development which includes an environment for the development of interactive 2D and 3D content including a rendering and physics engine and a scripting interface to program interactive content [30]. It allows the developers to export their applications into all the mainstream operation systems (Windows, Mac, Linux, IOS, Android)

through multiple platforms (desktop, web, mobile). Compared with traditional CAD software which always requires training and practice to get started, this game-based environment is more accessible for non-engineers to use.

V. USE CASE

Bolts and nuts are the most basic components in mechanical design. This study uses the modelling of a bolt to validate the proposed method and framework. This use case shows how the non-geometry knowledge can be captured in the knowledge capture environment and then visualised in the game-based product modelling environment.

In this research, a knowledge capture tool (Fig. 3) has been developed that allow designers to input the non-geometric information that are utilized in the product modelling process. Those non-geometric information are then decomposed into the VPM classes. When a bolt is being modelled in the traditional CAD software, the knowledge capture tool asks the designers to provide the associated knowledge at the same time. The captured knowledge is structured by the classified knowledge classes (example shown in Table I) and stored into a Knowledge File which is generated at the end of the knowledge capturing process (Fig. 3 (b)).

TABLE I
 KNOWLEDGE CAPTURE TEMPLATE WITH CAPTURED KNOWLEDGE OF A PRODUCT (MODELLING A BOLT)

Knowledge class	Captured knowledge
Name	Bolt
Category	Part
Function	Fasten
Form	Cylinder, external male thread
Behaviour	The bolt head locks the bolt in the place and nut is applied at the end.
Material	Steel
Rules	If bolt Length is 25mm, the hole depth should be 36mm.

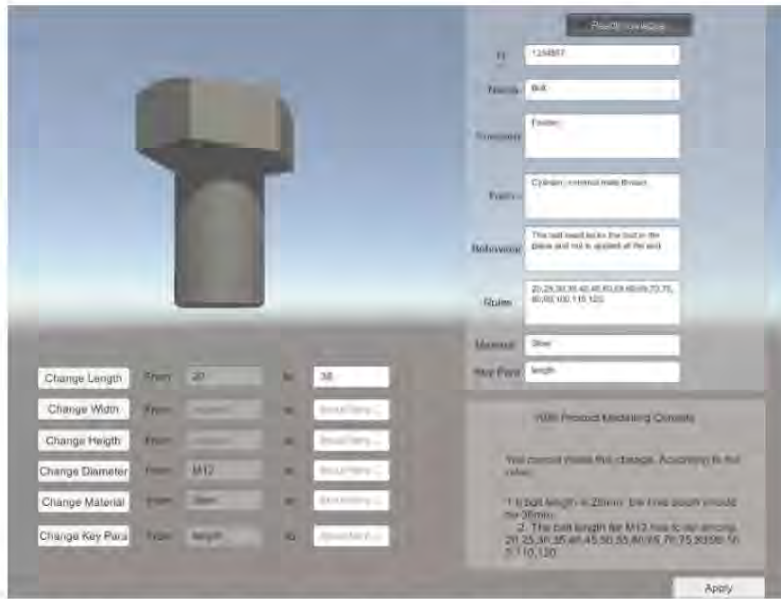


Fig. 4 The game-based product modelling environment developed in Unity3D

The developed game-based product modelling environment (Fig. 4) provides the ability to visualise the bolt's original geometry, as well as the possible changes that will happen and the associated knowledge that constrains the changes. The first type of visualisation is shown in the tool by importing the geometry file of the product into the interface through Unity3D plugin (Fig. 5 (a)). It provides the 3D view of the geometry of the product model. The second type of visualisation is the text representation of all the non-geometric information that is stored in the knowledge file (Fig. 5 (b)). This process is performed with the self-written code that automatically analyses and displays all the acquired non-geometric information that is stored in the knowledge file. The third type of visualisation is the possible change of geometry (Fig. 6 (a)). However, due to the nature of STEP, there are limited tools that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment (tested implementation tools shown in Table II). To develop such a method that can manipulate and visualise STEP files is beyond the scope of this research. Instead of

visualising the graphical changes in the geometry, this research uses text representation to indicate the changes that are made to the geometry.

TABLE II
TESTED IMPLEMENTATION TOOLS (TOOL VERSION UP TO 2020)

Tools	STEP Edit	Geometry View	User Interface Development
SolidWorks	x	✓	x
CATIA	x	✓	x
Siemens NX	x	✓	x
Autodesk Inventor	x	✓	x
Unity3D	x	✓(Plugin)	✓
Creo	x	✓	x
CAD Exchanger	x	✓	x
STP viewer	x	✓	x
IDA-STEP	x	✓	x
CAD Assistant	x	✓	x
Free CAD	x	✓	x



(a)

ID	t25467
Name	Bolt
Function	Fastener
Form	Cylinder, external hex, thread
Behaviour	The bolt head locks the bolt in the place and not to adjust it the etc.
Rules	1. If bolt length is 25mm, the hole depth should be 36mm. 2. The bolt length for M12 has to be among 20,25,30,35,40,45,50,55,60,65,70,75,80,90,100,110,120.
Material	Steel
Key Para	length

(b)

Fig. 5 (a) 3D view of the bolt geometry (b) visualisation of all the non-geometric information stored in the bolt knowledge file

Change Length	From: 20	To: 36
Change Width	From: <input type="text"/>	To: <input type="text"/>
Change Height	From: <input type="text"/>	To: <input type="text"/>
Change Diameter	From: M12	To: <input type="text"/>
Change Material	From: Steel	To: <input type="text"/>
Change Key Para	From: length	To: <input type="text"/>

(a)

NBE Product Modeling Console

You cannot make this change. According to the rules:

- If bolt length is 25mm, the hole depth should be 36mm.
- The bolt length for M12 has to be among 20,25,30,35,40,45,50,55,60,65,70,75,80,90,100,110,120.

Apply

(b)

Fig. 6 (a) Functions of making possible changes of bolt geometry (b) visualisation of the associated knowledge that are applied to constrain the change of geometry

```
#132=#133,#134,#135,#136,#137,#138,#139));
#95=TOROIDAL_SURFACE('',#372,2.613,0.23);
#96=CONICAL_SURFACE('',#360,4.,60.00000000000001);
#97=CONICAL_SURFACE('',#364,4.,60.00000000000001);
#98=CONICAL_SURFACE('',#364,4.,60.00000000000001);
#99=CONICAL_SURFACE('',#366,4.,60.00000000000001);
#100=CONICAL_SURFACE('',#366,4.,60.00000000000001);
#101=CONICAL_SURFACE('',#370,4.,60.00000000000001);
#102=CONICAL_SURFACE('',#373,2.1717,45.);
#103=CYLINDRICAL_SURFACE('',#356,2.413);
#104=FACE_BOUND('',#165,-1.);
#105=FACE_BOUND('',#170,-1.);
#106=FACE_BOUND('',#171,-1.);
#107=FACE_BOUND('',#172,-1.);
#108=FACE_BOUND('',#181,-1.);
#109=FACE_BOUND('',#190,-1.);
#110=FACE_BOUND('',#183,-1.);
#111=FACE_BOUND('',#184,-1.);
#112=CIRCLE('',#352,2.613);
#113=CIRCLE('',#354,2.613);
#114=CIRCLE('',#355,2.413);
#115=CIRCLE('',#357,2.1717);
#116=CIRCLE('',#359,4.);
#117=CIRCLE('',#361,4.);
#118=CIRCLE('',#363,4.);
#119=CIRCLE('',#365,4.);
#120=CIRCLE('',#367,4.);
#121=CIRCLE('',#369,4.);
#122=ADVANCED_FACE('',(#149),#140,-1.);
#123=ADVANCED_FACE('',(#150),#141,-1.);
#124=ADVANCED_FACE('',(#151),#142,-1.);
```

Fig. 7 Part of the STEP file of a bolt model that exported from NX10.0

The final stage of visualisation is to show the associated knowledge that are applied to make or constrain the change of geometry (Fig. 6 (b)). In this use case, the STEP file (example shown in Fig. 7) of the bolt model that exported from the CAD software and the Knowledge File (Fig. 3 (b)) from the knowledge capturing environment are loaded into game-based product modelling environment. When the designer is applying the changes of the length of the bolt, the environment will indicate the acceptable the length values according to the captured rules from the experienced designers (Fig. 6). In this way, new designers are guided by the knowledge to perform modelling activities.

VI. DISCUSSION

The method proposed in this paper is aimed at providing a generative product model as a generative representation of a product, which consists of the geometry information and non- geometry information, through a product modelling framework. The presented method for capturing the knowledge from the designers through a knowledge file provides a simple and efficient way in collecting and transferring knowledge. Further, the knowledge schema provides a clear view and format on the data that needed to be gathered in order to achieve a unified knowledge exchange between different platforms.

One of the advantages of the VPM is the explicit

classification of knowledge that required to in product design. The geometry information of a product is stored in VPM through a STEP file which has already been an interoperable format among CAD software. Hence, the proposed VPM can provide a generic product representation which enables seamless and cross-platform model communication.

A few limitations of the proposed method are discussed next. First, the designers need to spend extra time to use Knowledge Capture Tool, compared with modelling in traditional environment. This mainly happens at the first time when designers start to model without having a knowledge file. However, it is necessary to spend such time in order to capture the knowledge of designers and to build the knowledge repository. Secondly, functionality of modelling geometry in the developed game-based product modelling environment is limited. Due to the natural of STEP, there are no existing technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. In this game-based product modelling environment, the change of geometry mainly relies on the text description and knowledge indication. But this will not restrict the capability of the proposed modelling method. New standards and advanced geometry editing tools will accelerate the formation of a well-developed product modelling environment in the future.

VII. CONCLUSION

Design automation is a crucial demand from the industry to achieve a reduction in time and manpower cost of a product design. Capture, store, share and reuse the knowledge of experienced designers through a generic product model turns-out to be a challenge in product modelling. In order to provide non-engineers a product modelling environment where the existing knowledge can be used to guide and support their product modelling, a generative product model that integrates the geometry information with the associated knowledge that applied by experienced designers is required. The product model must have well-defined classes that can represent the entire product information and its associated knowledge. Also, the

product model needs to be represented in an interoperable schema to ensure a steady data exchange between different product modelling platforms. This paper has presented a VPM which consists of geometry and non-geometry information of a product. Further, this work presented a knowledge-based product modelling framework and explained how that captured data from a use case are modelled and visualised for enhancing the re-use of knowledge in product modelling. The future work includes improvement of the knowledge capture environment and further development of geometry editing functionality in the knowledge-based product modelling environment. As a continuation of the use case presented in this paper, a more complex product model will be tested in enhancing the performance of the proposed framework.

REFERENCES

- [1] Y. Haik and T. Shahin, *Engineering Design Process - second edition*, no. December. 2011.
- [2] C. X. Feng, C. C. Huang, A. Kusiak, and P. G. Li, "Representation of functions and features in detail design," *CAD Computer Aided Design*, vol. 28, no. 12, pp. 961–971, 1996, doi: 10.1016/0010-4485(96)00027-9.
- [3] F. P. Tolman, "Product modelling standards for the building and construction industry: past, present and future," *Autom. Constr.*, vol. 8, no. 3, pp. 227–235, 1999, doi: 10.1016/S0926-5805(98)00073-9.
- [4] W. Z. Yang, S. Q. Xie, Q. S. Ai, and Z. D. Zhou, "Recent development on product modelling: a review," *Int. J. Prod. Res.*, vol. 46, no. 21, pp. 6055–6085, 2008, doi: 10.1080/00207540701343895.
- [5] V. Shapiro, "Solid Modeling," *Handb. Comput. aided Geom. Des.*, vol. 20, pp. 473–518, 2002, doi: DOI: 10.1016/B978-044451104-1/50021-6.
- [6] I. Stroud, *Boundary Representation Modelling Techniques*, 1st ed. Springer-Verlag London, 2006.
- [7] Y.-M. Chen and C.-L. Wei, "Computer-aided feature-based design for net shape manufacturing," *Comput. Integr. Manuf. Syst.*, vol. 10, no. 2, pp. 147–164, 1997, doi: 10.1016/S0951-5240(97)00006-2.
- [8] O. W. Salomons, F. J. A. M. van Houten, and H. J. J. Kals, "Review of research in feature-based design," *J. Manuf. Syst.*, vol. 12, no. 2, pp. 113–132, 1993, doi: 10.1016/0278-6125(93)90012-I.
- [9] Michael J. Pratt, "Synthesis of an optimal approach to form feature modelling," in *Proceedings of the 1988 ASME International Computers in Engineering Conference and Exhibition*, 1988, vol. 1, pp. 263–274.
- [10] L. Wingård, "Introducing form features in product models: a step towards CAD/CAM with engineering terminology," PhD Dissertation, *Computer System for Design and Development*, 1991.
- [11] W. Van Holland and W. F. Bronsvort, "Assembly

- features in modeling and planning,” *Robot. Comput. Integr. Manuf.*, vol. 16, no. 4, pp. 277–294, 2000, doi: 10.1016/S0736-5845(00)00014-4.
- [12] W. Lawrence, “Using Knowledge-Based Engineering,” *Production*, p. 74, 1989.
- [13] A. Jurit H., Saia, A. and De Pennington, “Reasoning about machining operations using feature-based models,” *Int. J. Prod. Res.*, vol. 28, pp. 153–171, 1990.
- [14] L. W. Rosenfeld, “Solid modeling and knowledge-based engineering,” in *Handbook of solid modeling*, McGraw-Hill, Inc. New York, USA, 1995, pp. 91–911.
- [15] E. J. Reddy, C. N. V. Sridhar, and V. P. Rangadu, “Knowledge Based Engineering: Notion, Approaches and Future Trends,” *Am. J. Intell. Syst.*, vol. 5, no. 1, pp. 1–17, 2015, doi: 10.5923/j.ajis.20150501.01.
- [16] I. O. Sanya and E. M. Shehab, “An ontology framework for developing platform- independent knowledge-based engineering systems in the aerospace industry,” *Int. J. Prod. Res.*, vol. 53, pp. 1–27, 2014, doi: 10.1080/00207543.2014.965352.
- [17] E. M. Shehab and H. S. Abdalla, “Manufacturing cost modelling for concurrent product development,” *Robot. Comput. Integr. Manuf.*, vol. 17, no. 4, pp. 341–353, 2001, doi: 10.1016/S0736-5845(01)00009-6.
- [18] I.-S. Fan and P. Bermell-Garcia, “International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management,” *Concurr. Eng.*, vol. 16, no. 4, pp. 271–277, 2008, doi: 10.1177/1063293X08100027.
- [19] M. Cederfeldt, F. Elgh, and I. Rask, “A Transparent Design System for Iterative Product Development,” *J. Comput. Inf. Sci. Eng.*, vol. 6, pp. 300–307, 2006.
- [20] A. B. F. Kc Morris, STEP, the grand experience. Gaithersburg: National Institute of Standards and Technology, 1999.
- [21] S. Suresh, C. Egbu, and B. Kumar, “Key issues for implementing knowledge capture initiatives in small and medium enterprises in the UK construction industry,” no. January, 2006.
- [22] F. A. Salustri, “A formal theory for knowledge-based product model representation,” *Manuf. Syst.*, no. 519, pp. 1–19, 1996.
- [23] S. J. Fenves, “A core product model for representing design information,” *Tech. Rep. No. NISTIR 6736*, Natl. Inst. Stand. Technol., 2001, [Online]. Available: <http://www.mel.nist.gov/msidlibrary/doc/ir6736.pdf>.
- [24] F. Wang et al., “Towards modeling the evolution of product families,” *ASME Comput. Inf. Eng. Conf.*, 2003.
- [25] K. W. L. Mehmet Murat Baysal, Utpal Roy, Rachuri Sudarasan, Ram D. Sriram, “Product information exchange using Open assembly model: issues related to representation of geometric information,” 2005.
- [26] Z. Lou, H. Jiang, and X. Ruan, “Development of an integrated knowledge-based system for mold-base design,” *J. Mater. Process. Technol.*, vol. 150, no. 1–2, pp. 194–199, 2004, doi: 10.1016/j.jmatprotec.2004.01.037.
- [27] G. La Rocca, L. Krakkers, and M. J. L. van Tooren, “Development of an ICAD Generative Model for Blended Wing Body Aircraft Design,” in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 4-6 September 2002, Atlanta, Georgia, 2002, no. September, pp. 1–10, doi: 10.2514/6.2002-5447.
- [28] J. Groß and S. Rudolph, “Generating simulation models from UML - A FireSat example,” in *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, 2012, pp. 1–8.
- [29] J. Gross, A. Reichwein, S. Rudolph, D. Bock, and R. Laufer, “An Executable Unified Product Model Based on UML to Support Satellite Design,” in *Proceedings of the AIAA SPACE Conference*, 2009, no. September, doi: 10.2514/6.2009-6642.
- [30] Unity Technologies, “Unity User Manual,” Unity Documentation, 2020. <https://docs.unity3d.com/Manual/index.html> (accessed Jun. 18, 2020).

Enhancing Engineering Product Design using a Knowledge Based Game Engine Platform

Guolong Zhong*, Venkatesh Chennam Vijay, Noel Perera

Birmingham City University, Faculty of Computing, Engineering and the Built Environment, Millennium Point, Curzon Street, Birmingham, West Midlands, B4 7XG, United Kingdom

*Corresponding author.

E-mail: guolong.zhong@mail.bcu.ac.uk, venkatesh.vijay@bcu.ac.uk, noel.perera@bcu.ac.uk

Abstract

Traditional CAD tools and systems cannot explain the real-world concepts by themselves and require the users to have knowledge and design experience of the product in order to understand design rules and judge the correctness of the changes. Knowledge-based engineering (KBE) has been introduced to address these issues; however, existing KBE methodologies offer limited instantiation steps and enabling tools for implementation. In this paper, a knowledge-based product modelling prototype system is proposed to overcome the above problems. This system is developed based on the Virtual Product Modelling framework using a game engine platform to aid in capturing, reusing, and exchanging the existing product information and provide knowledge reasoning in the product modelling process. Three different use cases were applied in the research to validate the effectiveness of the proposed system. The use case evaluation has proved that the developed prototype system can help save time and prevent engineers from making mistakes in the product design. The findings of this research have shown the potential of integrating product modelling with VR/AR visualisation techniques and applying this system to non-engineers.

Keywords:

Product design; Knowledge-based engineering; Game engine; Design rules

1. Introduction

1.1 Background

The rapid development of science and information technologies has generated higher demands for industrial development capacity, productivity and agile response to the market. They all drive the industry to design and produce more complex products at lower costs and with less time. Computer Aided Design (CAD) has been introduced as a Design Engineering Automation (DEA) method for completing product design. Nonetheless, CAD tools and systems cannot understand and explain real-world design concepts by themselves. To judge the correctness of the design, CAD tools and systems require users to have sufficient knowledge and design experience of the product. The trend of product design has evolved from CAD to Computer Aided Product Modelling and then to Knowledge Based Product Modelling. This has required the product design software and environment to provide more interaction between end-users and the product modelling process through the reuse of existing knowledge to support the product modelling. However, the knowledge of a product is often discrete, unorganised and in various forms [1]. In a traditional product design environment, the designer can only create and provide the visible geometry information in the product model. Only geometry information is not enough to describe a product model completely. There is a lack of required knowledge representation in multiple aspects of product design.

Existing research works show that there are two challenges in integrating extra data with CAD models: capturing and managing the product data in the complex product models using traditional CAD [2]–[6] and the mismatch between the availability of information and the accessibility of the appropriate information to designers [7]. Thus, there is a need to develop a product model which can capture and reuse complex product data and provide accessible and appropriate information to designers.

In the last 20 years, Knowledge Based Engineering (KBE) has shown its advantages in product development in different engineering areas such as automation, mechanical engineering, civil engineering and aerospace engineering in terms of modelling and cost reduction. It helps automate the repetitive design tasks by capturing, integrating, utilising and reusing existing knowledge required in various aspects of the product design [8], [9]. The use of KBE methods and techniques has played an important role in design engineering automation for the development of a product in the industry [10], [11]. The product data involved in product design has been expanded from “geometry-only” to “knowledge-integrated”. However, existing product models and CAD tools show limited capabilities in capturing and reusing knowledge. Further, the substantiation steps for implementing the current KBE methods and techniques for product modelling are usually not available and understandable to users [12]–[14]. Therefore, to overcome this “black-box” problem, it is necessary to develop a KBE implementation framework along with use cases and enabling tools for the purpose of capturing and reusing design knowledge in product modelling for design engineering automation.

Capturing and transferring the knowledge of experienced designers are difficult. According to [1], the main challenge existing in small and medium companies and industries for implementing knowledge capture initiatives is a lack of awareness of knowledge capture benefits. Due to this, individuals and small and medium enterprises lack the vision and strategy and structure for knowledge capture. They have a strong reliance on informal networks and collaboration to locate the repository of knowledge.

Therefore, a critical issue of knowledge-based product modelling is capturing, classifying, structuring, and managing the captured knowledge. Since there is no clear formalised link between a generic product model and an interoperable format in the KBE environment, it is essential to provide well-defined knowledge classes and a formalised knowledge capture

method for individuals, enterprises and industries to capture and share knowledge instead of using informal oral communication or notes and spreadsheets in different formats. In this paper, geometric data contained in a CAD file is regarded as “geometry information”, and non-geometric information, such as experience, expertise and design rules, is called “knowledge”.

1.2 Knowledge Reasoning and Rules

A knowledge reasoning engine can be used to realise the inference mechanism to provide the “How” and the “Why” for solving a posed problem in the industry [15]. As explained by [16], the knowledge reasoning approach in the inference mechanism can be performed by selecting, using and matching various rules. It is important to define rules within the model to provide an interpretation for both humans and computers to avoid misunderstandings and misuse of product models [17].

The rules are usually defined through the use of object-oriented language to handle semantics and internal relations within the model. Engineering rules are often formed from product design and process knowledge [18]. Therefore, it can be seen that design rules from the existing product design knowledge can be used to build the interaction between geometry and design information and thus provide the knowledge reasoning. With the application of rules, knowledge reasoning can be performed in an interactive application.

1.3 Interactive application development using a game engine

The development of an interactive application consists of two main components: the application and the content [19]. The application aims to provide information in real-time to end-users and the ways to interact with it. The content contains the information through which the application navigates and provides a view to the users.

The announcement of the COLLADA [20] format becoming an ISO standard in industrial automation systems and integration shows the potential of adopting elements from the gaming industry in Engineering [21]. According to the interactive qualifying project report [22], developers have recently realised that game engines can be successfully used for non-game applications development, such as architecture prototyping, interactive applications and research data visualisation. Gaming engines are generally used as an integrated development environment to enable the rapid development of game applications and build interactive applications. Although several game engines are in the market, Unreal and Unity are the two main game engines widely used in the current gaming industry.

Unreal was first released in 1995 and has become one of the major game engines being employed in the gaming industry. The Unreal game engine features fast rendering and high-quality graphics; hence it is preferred for building large games. Unity is a cross-platform game engine that was announced in 2005. It has become more popular and adopted by a growing number of users in recent decades due to its easy accessibility, user development support and strength in making 2D and 3D simulations [23]. Apart from the gaming industry, Unity is also used by industries such as automotive, architecture, engineering, and construction [24], [25]. A survey [26] showed that Unity has been playing an active role in the game development field, and the usage and industrial devotion to Unity is increasing and amplifying.

This research aims to use a game engine platform to develop a knowledge-based product modelling system that could capture, reuse and integrate the associated knowledge applied by designers to provide interactions with knowledge reasoning and to enhance the product design process. The rest of the paper will explain the overall implementation methods and discuss in detail how the design engineer's knowledge is captured, reused, and mapped into

the system in each use case to enhance the product modelling process for design engineering automation.

2. Methods

A Virtual Product Modelling (VPM) framework for developing a knowledge-based product modelling environment that enables existing knowledge to be captured and reused in product modelling has been developed and discussed in a previously published paper by the authors [27]. This framework (see Figure 1) consists of the following five stages:

- 1) Product model development
- 2) Knowledge capture of non-geometric information
- 3) Knowledge capture of geometry information
- 4) Knowledge mapping
- 5) Product visualisation and validation

This framework provides a set of activities for design engineers to conduct for implementing the knowledge-based engineering technique in the modelling process. It also provides the ability for parametric geometry representation as components are associated with parametric values after the implementation.

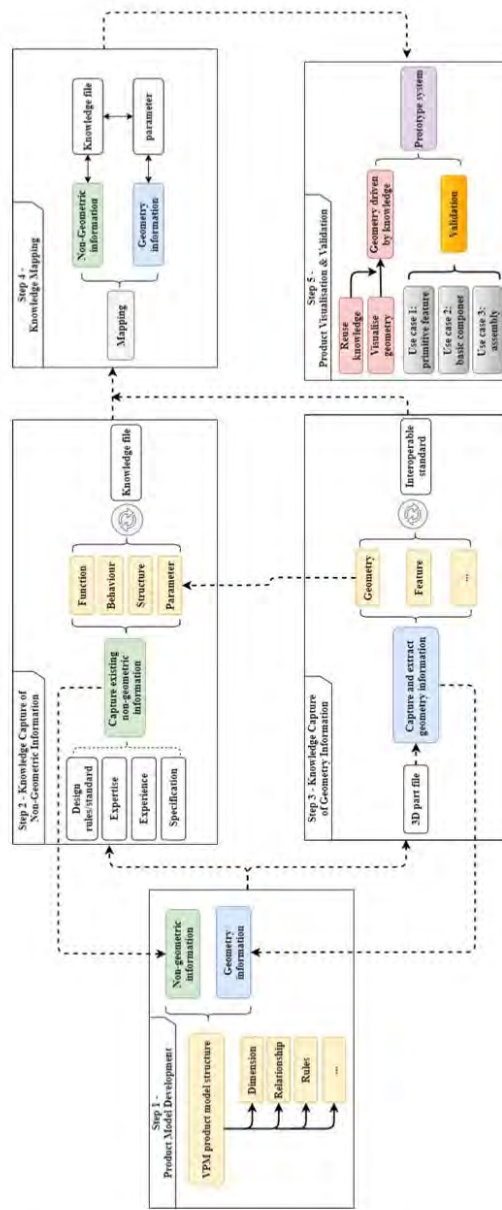


Figure 1: Virtual product modelling framework [27] (print in colour)

2.1 Product model development

In the first stage, a generic product model structure that can represent all required design information of the product will be developed using the meta class of VPM. The product model structure (as shown in Figure 2) is a comprehensive and generative representation of the product with all its essential non-geometric and geometric information. It provides a data structure that includes all blocks describing a product from different aspects. Moreover, this hierarchical product structure also possesses inheritance characteristics as it is developed with the object-oriented concepts in UML. A generic product model will be formed by integrating the knowledge captured in the second and third stages into this product model structure.

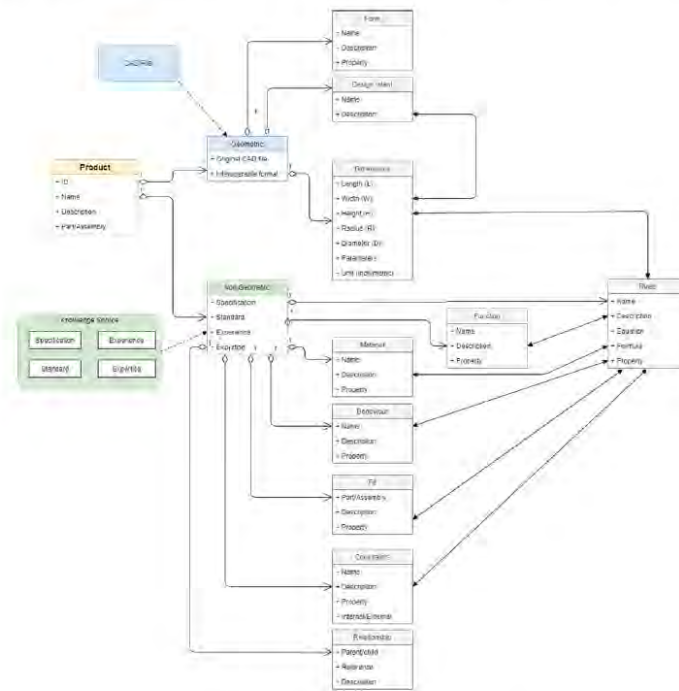


Figure 2: Developed product model structure from VPM (print in colour)

2.2 Knowledge capture of non-geometric information

The second stage of the methodology framework is to capture the existing non-geometric knowledge of the product. It includes identifying essential aspects that should be considered in the product modelling process, defining meta classes for each non-geometric aspect, capturing the existing non-geometric information and breaking the non-geometric information down into the classified components. Tasks that need to be performed in this stage are listed below (also shown in Figure 3):

1. Identify the essential non-geometric information used to specify the product from the existing design knowledge.
2. Decompose the non-geometric information into atomic classes.
3. Identify the rules that are applied in the design process of the product. Then the design engineer can then model the interactions between the non-geometric information and geometry.
4. Export the non-geometric information into a generalised format that can be reused and transferred between different industry APIs.

After performing these tasks, a knowledge file containing well-decomposed non-geometric knowledge classes and properties is generated. This knowledge file represents the product from non-geometric aspects and can be used as a knowledge base to share the product's information between design engineers. It can be further utilised to enhance product modelling by visualising the knowledge in a knowledge-based product modelling environment.

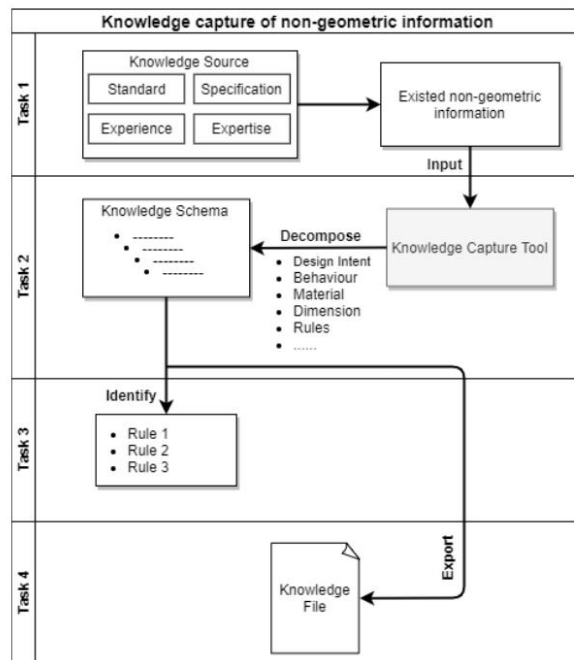


Figure 3: Task flow of capturing the non-geometric information

2.3 Knowledge capture of geometry information

The next step in the virtual product modelling framework is to capture the geometric information of the product. In Design Engineering Automation, the geometric information of a product model can be automatically saved into a digital part file in the current CAD platform/software. An international standard needs to be followed as an interoperable format to allow geometric data exchange among different CAD platforms/software and avoid data conversion faults. The STEP standard has been identified and used in this research as an interoperable format for exchanging product geometric information among different CAD platforms/software. Tasks performed in this stage are shown below (also see Figure 4):

1. Model the product geometry in the CAD platform/software.
2. Export the product CAD model into an interoperable and standardised format.

The result of performing these tasks is the geometry file with the geometric data that describes the product's geometry and can be used to exchange among different CAD platforms/software.

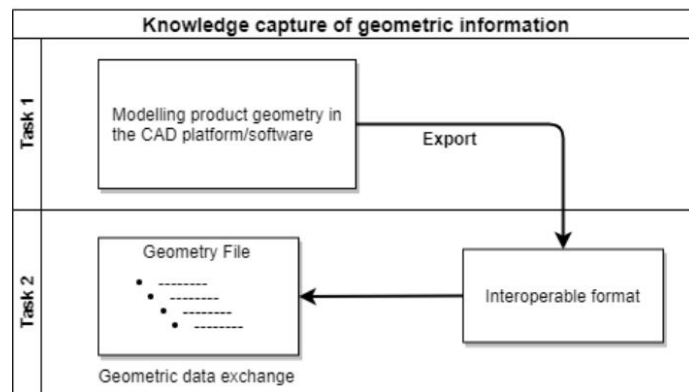


Figure 4: Task flow of capturing the geometric information

2.4 Knowledge Mapping

Knowledge Mapping is the most significant stage of the framework, which build the connection and interaction between the non-geometric information and geometry. In this stage, non-geometric information is transferred from the knowledge file to an object-oriented programming environment, where key parameters and design rules are linked to the product geometry. The data from the knowledge capture tool is exported into a knowledge file in an XML format and parsed using object-oriented programming. XML has been widely used as a generalised data format that is both human-readable and machine-readable for data exchange between different Application Programming Interfaces (APIs) in the industry. Since various knowledge classes and properties have been defined in the proposed virtual product modelling framework, object-oriented programming is employed as a programming paradigm that provides high modularity and reusability for these knowledge classes and properties.

Further, a knowledge mapping framework has been developed (as shown in Figure 5) to explain how knowledge reasoning is performed in this knowledge mapping stage.

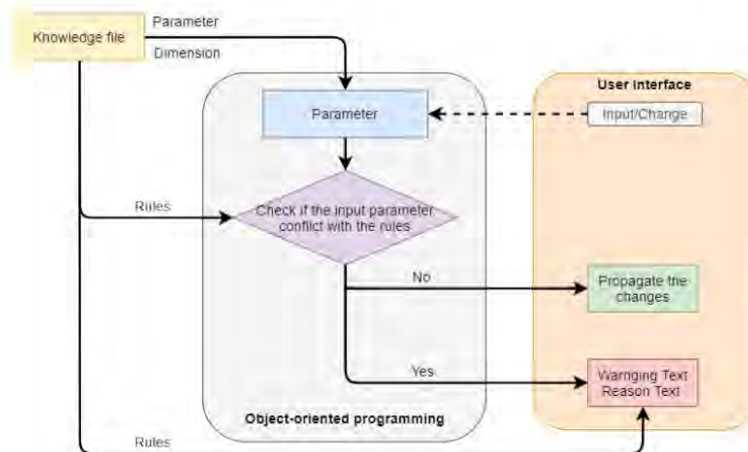


Figure 5: Knowledge mapping framework for knowledge reasoning (*print in colour*)

2.5 Product visualisation and validation

The last step is the visualisation and validation stage, where the developed product model in VPM is visualised and checked for the correct representation of the initial model. Visualisation is an integral part of the proposed framework as it provides a straightforward and effective way of understanding the product model. The developed framework was applied to a KBE product modelling prototype system developed in Unity 3D software. Unity 3D has a license-free version for personal development, which includes an environment for the development of interactive 2D and 3D content, including a rendering and physics engine and a scripting interface to program interactive content [28]. It allows the users to export their applications into all the mainstream operation systems (Windows, Mac, Linux, IOS, Android) through multiple platforms (desktop, web, mobile). Testing use cases are performed to identify and evaluate the proposed framework's effectiveness. Suppose the design engineer changes one dimension of the geometry of the initial product model; in that case, the virtual

changes one dimension of the geometry of the initial product model; in that case, the virtual product modelling framework will check the rules that determine this geometry and propagates the rules and changes to the design engineer. In this case, design engineers will know what will be affected if the geometry is changed in this model and the constraints of these changes. The visualisation of a hex bolt model example in Figure 6 includes: (a) a 3D view of the bolt geometry and (b) visualisation of all the non-geometric information stored in the bolt knowledge file, (c) Functions of making possible changes to bolt geometry, (d) visualisation of the associated knowledge that is applied to constrain the change of geometry.

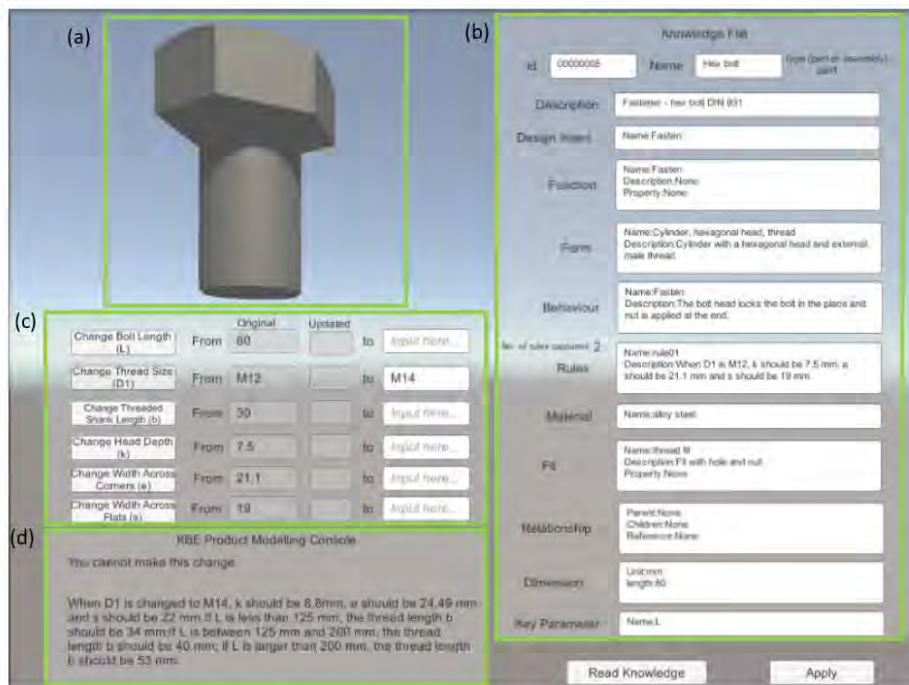


Figure 6: Example of visualisation in the developed knowledge-based product modelling environment (print in colour)

3. Implementation and Validation

The proposed system was validated by three use cases in this research. The first use case is adapted from the primitive design feature examples for the basic engineering feature modelling [29]. This use case is selected because primitive design features are the fundamental geometric features applied in the actual product modelling process in general CAD environments. The second use case is a hex bolt example from literature. Since the bolt is the most basic component for mechanical engineering, modelling this component is supposed to describe how engineering rules are applied in the basic engineering part. The third use case selects a wheel assembly example from literature because it has been widely used as a demonstrative model to explain the model structure, component relationships and complex parameter configurations [30]. This wheel assembly with a wheel part and a tyre part is supposed to describe how two engineering parts are connected and constrained in one engineering assembly. The following subsections explain the validation through the implementation of each use case.

3.1 Use case 1

Primitive features are basic geometric features from which many other design features can be created. The basic primitive design features are block, cylinder, sphere and cone. Four parts with primitive design features are selected based on these features in this use case (as shown in Figure 7).

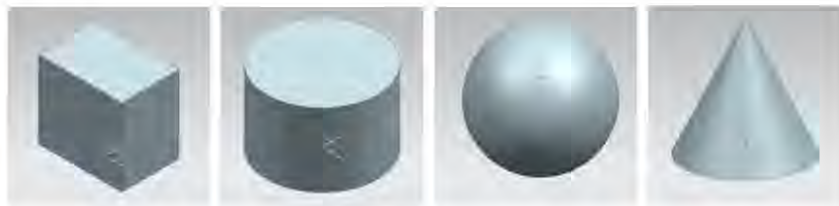


Figure 7: Four simple parts with primitive design features (modelled in Siemens NX 10)

(print in colour)

Existing information of use case 1 (example shown in Table 1) is collected from the part library of one of the current product modelling systems – Siemens NX 10. However, the Siemens NX part library does not provide all the information that fits the classified VPM classes. Hence, for some VPM classes, such as material, behaviour, fit, and relationship, the entities are given as “None” or “Not defined”.

Table 1: Existing information of use case 1- block part example

VPM knowledge class	Existing product information
Product	Block
Feature	Primitive design feature - block
Description	The Block is a cube.
Function	None
Behaviour	None
Form	Primitive design feature - block
Material	Not defined
Design intent	Primitive design feature to create other design features.
Geometry	From STEP file
Dimension	Length =100mm, width =100mm, height=100mm
Rules	Block Length L = Block Width W = Block Height H
Fit	None
Constraint	None
Relationship	None
Reference	None

The implementation follows the same process as discussed in Section 2. The function of making changes to product geometry is limited to basic parameters based on the use case. In use case 1, different simple parts are used to test the effectiveness of the methodology and the workability of each function. In the block part example, one testing scenario is identified - “single dimension changed (block length) with the single rule applied”. The design rule that has been captured and applied to the block part is “Length(L) = Width(W) = Height (H)”. In the object-oriented programming (*IF-THEN-ELSE* statement), this rule can be expressed as:

“IF block length is changed, *THEN* block width and block heights need to be changed;
ELSE IF block width is changed, *THEN* block length and height need to be changed;
ELSE IF block height is changed, *THEN* block length and width need to be changed.”

The resulted visualisation in the developed user interface is shown in Figure 8. After importing the STEP file of the block part into the user interface, a 3D model of the block can be visualised. By clicking the “Read Knowledge” button, the developed tool will automatically parse the knowledge file and display all the captured information in the interface. After the user selects to change the block length by clicking the button “Change Length” in the tool interface, the resulting changes will be shown in the “KBE Product Modelling Console” panel. For instance, when the user input 120 in the interface to change the length from the original 100 to 120 (unit: mm), the “KBE Product Modelling Console” will analyse if this change can be made by checking the rules. In this testing scenario, the rule that has been applied to the block part is “Length(L) = Width(W) = Height (H)”. Therefore, the width and height of the block are also changed to 120 automatically. And the affecting rule is shown correctly in the “KBE Product Modelling Console”, which fulfils the knowledge reasoning for this product modelling process.



Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box – button pressed to apply the change; Blue box - knowledge reasoning.

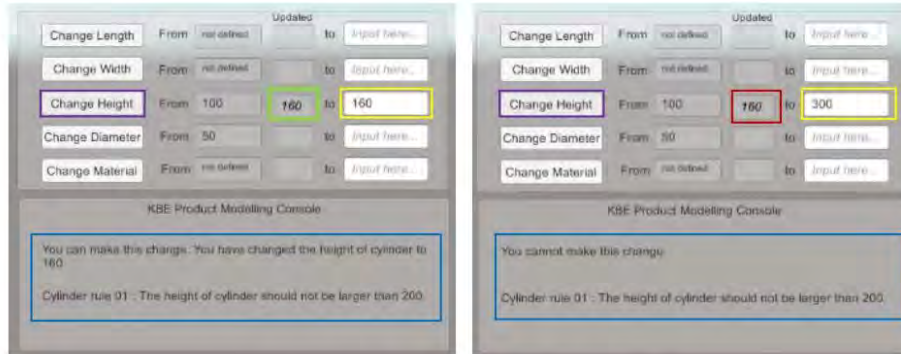
Figure 8: Results of validation – use case 1: Block part (print in colour)

Similarly, different testing scenarios have been defined to test the function of “Change Height”, “Change Diameter”, and “Change Material” by using cylinder part, cone part and sphere part. The rules applied in these testing scenarios are listed in Table 2.

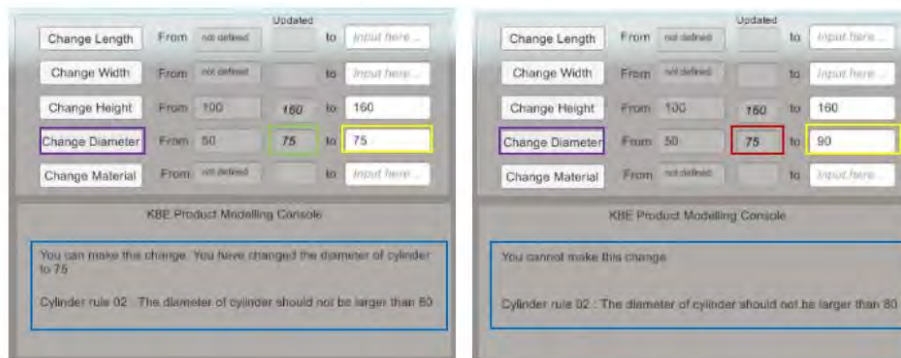
Table 2: Rules applied in the cylinder, cone, and sphere parts.

	Rules	Description
Cylinder	Cylinder rule 01	The height of cylinder should not be larger than 200.
	Cylinder rule 02	The diameter of cylinder should not be larger than 80.
Cone	Cone rule 01	The base diameter of cone should be 10,16,18,20 mm
	Cone rule 02	If the diameter of cone is less than 16mm, then the height should be 18mm. If the diameter of cone is equal to or larger than 16 mm, then the height should be 24 mm.
Sphere	Sphere rule 01	The diameter of sphere should be among 19,20,21,22,25,30,35,40 mm
	Sphere rule 02	The material of the steel ball should be among AISI 201, AISI 304, AISI 316 stainless steel.

The implementation processes remain the same as described in Section 2. The results of the validation are shown in figures 9, 10 and 11, respectively.



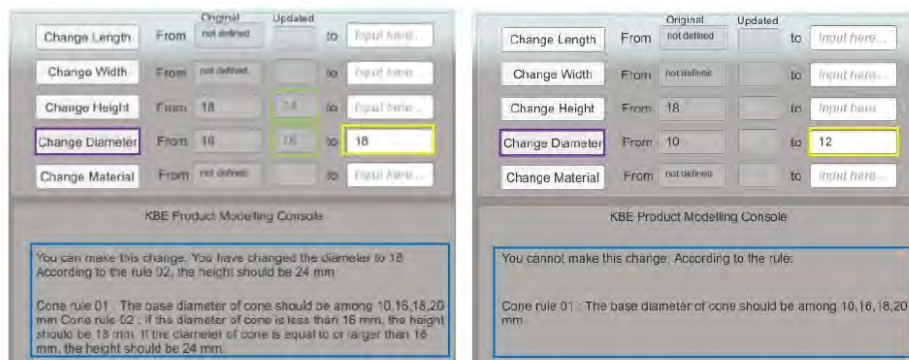
(a) Scenario One - change of height with rule 01



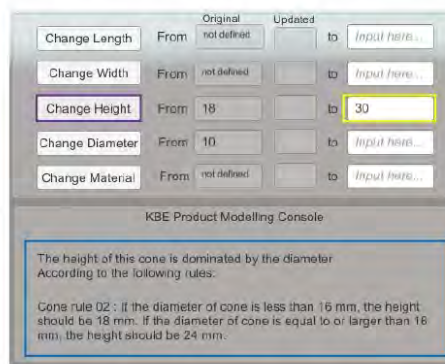
(b) Scenario Two - change of diameter with rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Red box - propagated parameter (changes not allowed by rules); Purple box - button pressed to apply the change; Blue box - knowledge reasoning.

Figure 9: Results of validation – use case 1: cylinder part (*print in colour*)



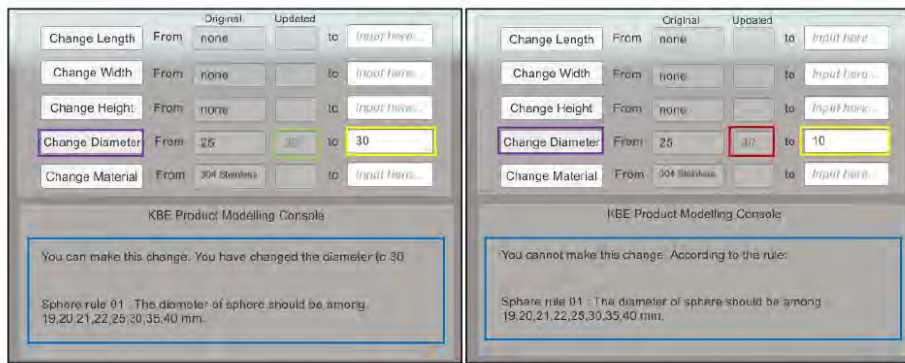
(a) Scenario One - change of base diameter



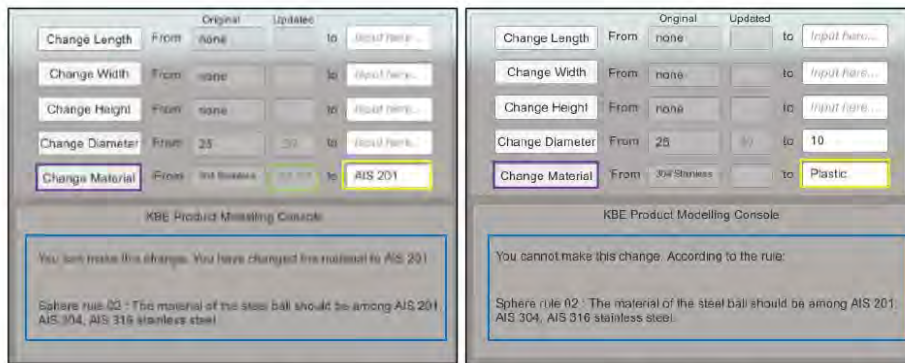
(b) Scenario Two - change of height against cone rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box - button pressed to apply the change; Blue box - knowledge reasoning.

Figure 10: Results of validation – use case 1: cone part (print in colour)



(a) Scenario One - change of diameter under rule 01



(b) Scenario Two - change of material under rule 02

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Red box - propagated parameter (changes not allowed by rules); Purple box - button pressed to apply the change; Blue box - knowledge reasoning.

Figure 11: Results of validation – use case 1: sphere part (*print in colour*)

3.2 Use case 2

The second use case selects a hex bolt that has more knowledge to evaluate the effectiveness of the developed prototype system in application to an engineering part with complex design rules. A hex bolt has a hexagonal head and external machine threads for a firm and rough handling. It is usually in a wide range of sizes for custom application on its dimensional requirements. The material of hex bolts varies from steel, alloy steel, carbon steel and anti-corrosion stainless steel,

depending on the different application environments. Figure 12 shows the 2D drawings and the dimensional descriptions of the hex bolt.

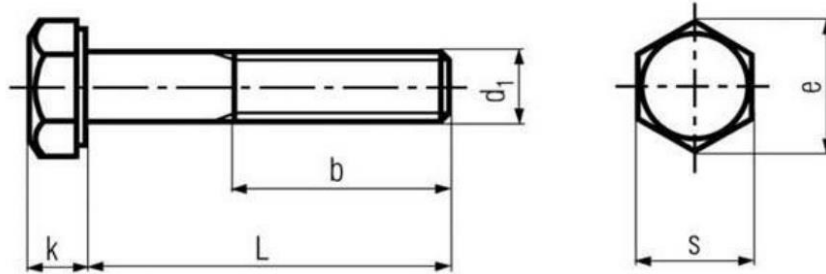


Figure 12: 2D drawings of the hex bolt with dimensional descriptions

As the hex bolt is a standardised part, existing product information can be collected from the existing hex bolt standard. Table 3 shows the dimensions of the hex bolt in the DIN 931 standard.

Table 3: Hex bolt dimensions (in millimetres) – DIN 931 (partial)

Thread size D1	Threaded shank length b (L* < 125)	Threaded shank length b (L - 125 to 200)	Threaded shank length b (L > 200)	Head depth k	Width across corner e	Width across flats s
M10	26	32	45	6.4	18.9	17
M12	30	36	49	7.5	21.1	19
M14	34	40	53	8.8	24.49	22
M16	38	44	57	10	26.75	24
M18	42	48	61	11.5	30.14	27
M20	46	52	65	12.5	33.53	30
M22	50	56	69	14	35.72	32
M24	54	60	73	15	39.98	36

*L: Bolt length

For this use case implementation, an M12 hex bolt is selected with a bolt length of 80 mm. According to DIN 931, the thread length b is 30 mm. If the D1 of this hex bolt is being changed, other dimensional parameters have to be modified based on the standard

correspondingly. Two rules that constrained dimensional parameters are extracted from the DIN 931 standard and defined as follows:

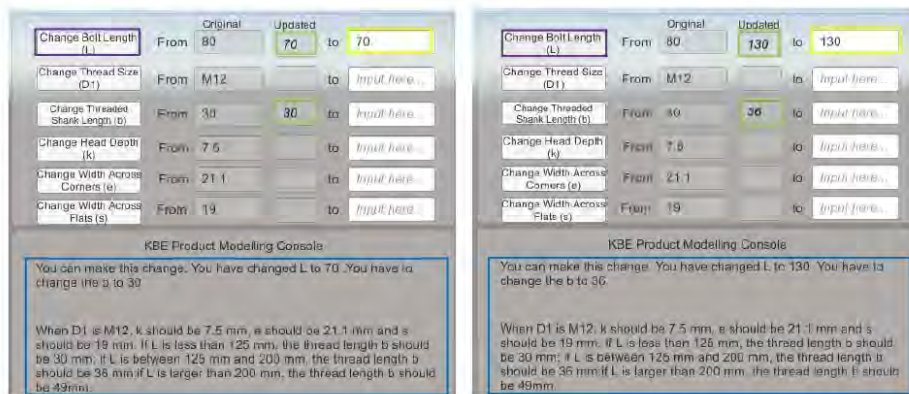
- Bolt rules 1 – when D1 is M12, k should be 7.5 mm, e should be 21.1 mm and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm.
- Bolt rules 2 – when D1 is changed to M14, k should be 8.8mm, e should be 24.49 mm and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.

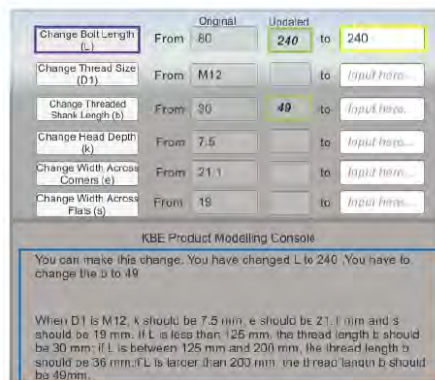
The existing information of an M12 hex bolt in this use case is shown in Table 4.

Table 4: Existing information of use case 2 – hex bolt

VPM knowledge class	Existing product information
Product	Hex bolt
Feature	hexagonal head and external male thread
Description	Fastener - hex bolt DIN 931
Function	Fasten
Behaviour	The bolt head locks the bolt in the place, and nut is applied at the end.
Form	Cylinder with a hexagonal head and external male thread
Material	Alloy steel
Design intent	Fasten
Geometry	From STEP file
Dimension	D1: M12, L=80 mm, b =30mm, k=7.5 mm, e =21.1 mm, s=19 mm.
Rules	Bolt rules 1 – when D1 is M12, k should be 7.5, e should be 21.1, and s should be 19 mm. If L is less than 125 mm, the thread length b should be 30 mm; if L is between 125 mm and 200 mm, the thread length b should be 36 mm; if L is larger than 200 mm, the thread length b should be 49mm. Bolt rules 2 – when D1 is changed to M14, k should be 8.8, e should be 24.49, and s should be 22 mm. If L is less than 125 mm, the thread length b should be 34 mm; if L is between 125 mm and 200 mm, the thread length b should be 40 mm; if L is larger than 200 mm, the thread length b should be 53 mm.
Fit	Fit with hole and nut.
Constraint	None
Relationship	None
Reference	DIN 931

Use case 2 is implemented in the same way as explained previously in Section 2. From the bolt rule 01, it can be known that the b value is constrained by the bolt length L and should be either 30, 36 or 49 for an M12 DIN 931 hex bolt. When the users change the b value in the user interface, the algorithm will check the b value against the bolt rule 01. If the input b value is among 30, 36 and 49, the user interface will tell the users that the bolt length L has to be changed accordingly. If the input b value is not in the given range from the bolt rule 01, the user interface will show that the value is conflicting with the rule through a warning text. When the user inputs D1 values, the algorithm will check the k, e, s values from the embed rules. If the users input k, e and s values, the algorithm will compare the input values with the values specified in the rules and then suggest if the changes can be made. Similarly, the algorithm will check the input L and b values with the rules, tell users whether these values are allowed by the rules, and show the relevant reasons why the changes cannot be performed. The validation is performed along with the process of visualisation, and the results are shown in Figure 13 and Figure 14.

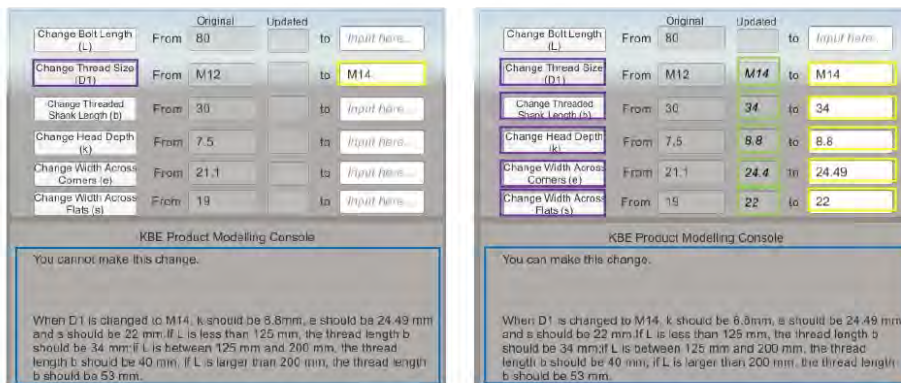




(c) $L > 200$ mm

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box - button pressed to apply the change; Blue box - knowledge reasoning.

Figure 13: Results of validation – use case 2: hex bolt, scenario one - change L



(a) b, k, e, s values conflict with rule

(b) b, k, e, s values allowed by rule

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box - button pressed to apply the change; Blue box - knowledge reasoning.

Figure 14: Results of validation – use case 2: hex bolt, scenario two – change bolt thread size

D1 from M12 to M14 (print in colour)

3.3 Use case 3

The implementation of a wheel assembly model in this research aims to validate that the proposed prototype system can also be applied to the engineering part with assembly relationships and internal and external parameter constraints. Figure 15 shows a wheel assembly modelled in Siemens NX software.



Figure 15: A wheel assembly modelled in Siemens NX 10 (*print in colour*)

In this use case, the wheel assembly consists of two basic parts: a tyre and a wheel. In real life, when the tyre is inflated, the air pressure within the tyre keeps the tyre bead in the groove of the wheel. Theoretically, when the wheel assembly is designed, the dimensions of the tyre need to fit with the sizes of the wheel to ensure that these two parts are correctly assembled. Thus, in this use case, the existing product information can be extracted from the above theoretical assembling relations and dimensional parameters of the tyre and wheel parts. The following figures 16 and 17 show the dimensional parameters of the wheel and tyre parts, respectively.

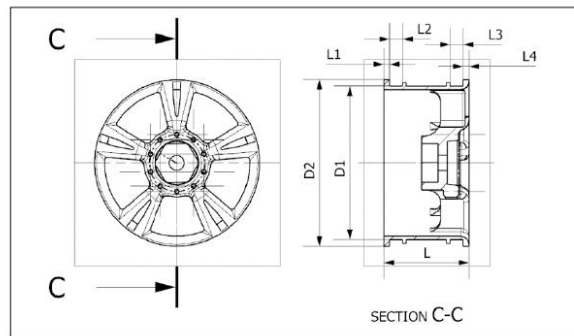


Figure 16: 2D drawings of the wheel part with dimensional descriptions

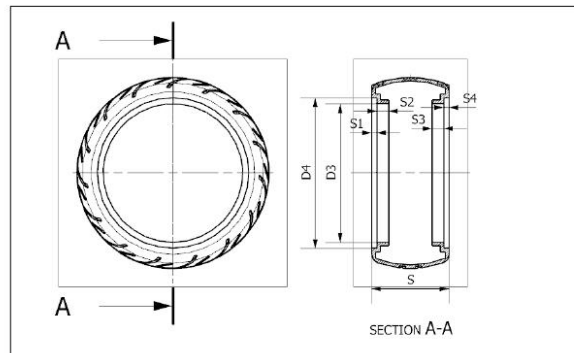


Figure 17: 2D drawings of the tyre part with dimensional descriptions

As mentioned previously in this section, when the wheel part and tyre part are assembled, all these assembling dimensions need to fit with each other. In this way, the external and internal constraints between parameters and rules of the wheel assembly are derived in Table 5 and Table 6.

Table 5: Wheel assembly parameters and constraints

Wheel Assembly	Parameters	Part External Constraints	Part Internal Constraints
Wheel	D1	D1=D3	-
	D2	D2=D4	-
	L	L=S	-
	L1	L1=S1	L1=L4
	L2	L2=S2	L2=L3
	L3	L3=S3	L3=L2
	L4	L4=S4	L4=L1
Tyre	D3	D3=D1	-
	D4	D4=D2	-
	S	S=L	-
	S1	S1=L1	S1=S4
	S2	S2=L2	S2=S3
	S3	S3=L3	S3=S2
	S4	S4=L4	S4=S1

Table 6: Wheel assembly rules

	Rules	Description
Wheel assembly	Assembly rules 1	D1 = D3, when D1 is changed, D3 needs to be changed as well, and vice versa.
	Assembly rules 2	D2 = D4, when D2 is changed, D4 needs to be changed as well, and vice versa.
	Assembly rules 3	L1 = S1, when L1 is changed, S1 needs to be changed as well, and vice versa.
	Assembly rules 4	L2 = S2, when L2 is changed, S2 needs to be changed as well, and vice versa.
	Assembly rules 5	L3 = S3, when L3 is changed, S3 needs to be changed as well, and vice versa.
	Assembly rules 6	L4 = S4, when L4 is changed, S4 needs to be changed as well, and vice versa.
	Assembly rules 7	L = S, when L is changed, S needs to be changed as well, and vice versa.
Wheel part	Wheel rules 1	L1 = L4, when L1 is changed, L4 needs to be changed as well, and vice versa.
	Wheel rules 2	L2 = L3, when L2 is changed, L3 needs to be changed as well, and vice versa.
Tyre part	Tyre rules 1	S1 = S4, when S1 is changed, S4 needs to be changed as well, and vice versa.
	Tyre rules 2	S2 = S3, when S2 is changed, S3 needs to be changed as well, and vice versa.

The existing information of the wheel assembly is collected and shown in Table 7. The visualisation (see Figure 18 and 19) and validation process are completed the same way as described previously.

Table 7: Existing information of use case 3 – wheel assembly

VPM knowledge class	Existing product information
Product	Wheel assembly
Feature	Circular component, assembly
Description	Assembly of a wheel part and a tyre part
Function	Mounting and rotating for movement
Behaviour	Keeps wheel attached to a hub and the car axle.
Form	Circular
Material	Mixed
Design intent	Mounting and rotating for movement
Geometry	From STEP file
Dimension	D1:1884 mm D2:2046 mm D3:1884 mm D4:2046 mm L:1045 mm S:1045 mm L1:70 mm L2:160 mm L3:70 mm L4:160 mm S1:70 mm S2:160 mm S3:70 mm S4:160 mm
Rules	Wheel assembly rules 1,2,3,4,5,6,7
Fit	Tyre fits with wheel
Constraint	Assembly constraints
Relationship	Parent of the wheel part and tyre part
Reference	None

When the users change the L1 parameter of the wheel part, the tool interface shows that the S1 parameter in the tyre part needs to be adjusted based on the wheel assembly rule 03. In the meantime, the L4 parameter in the wheel part needs to be changed according to the wheel part rule 01, and the S4 needs to be changed according to the tyre rule 01. Therefore, only when all the required changes are applied will the tool allow the change of the L1 parameter to be made by the users. The validation results are shown in Figure 20.

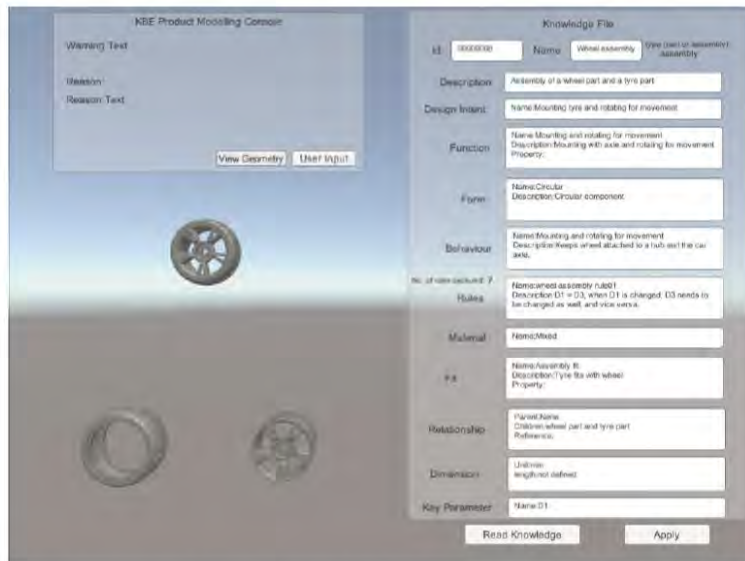


Figure 18: Wheel assembly model visualised in the developed knowledge-based product modelling environment (*print in colour*)

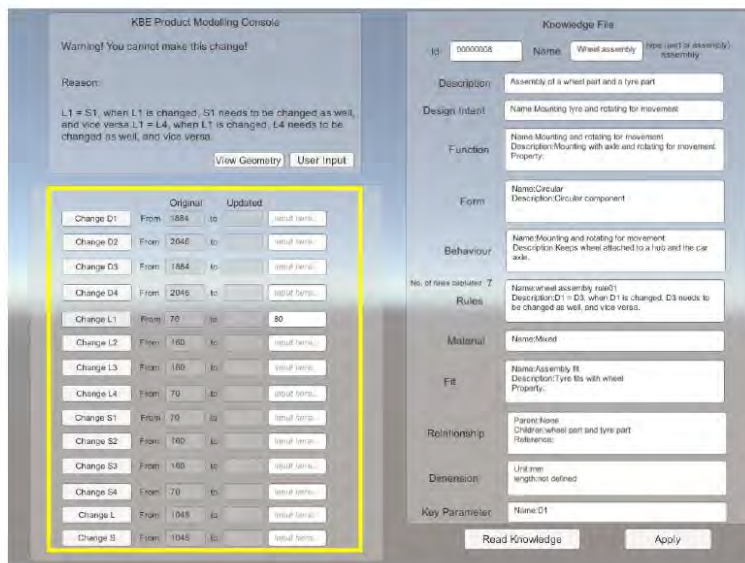
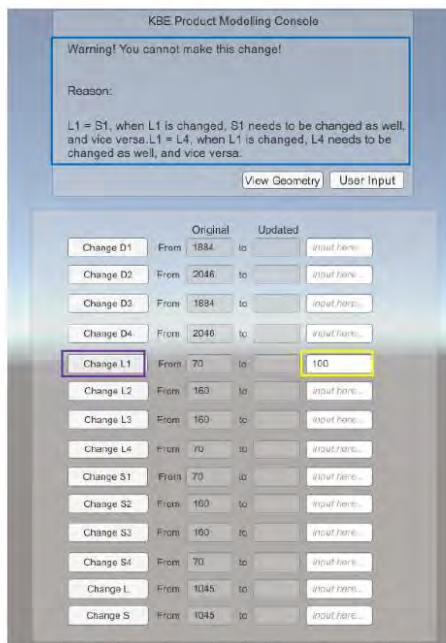
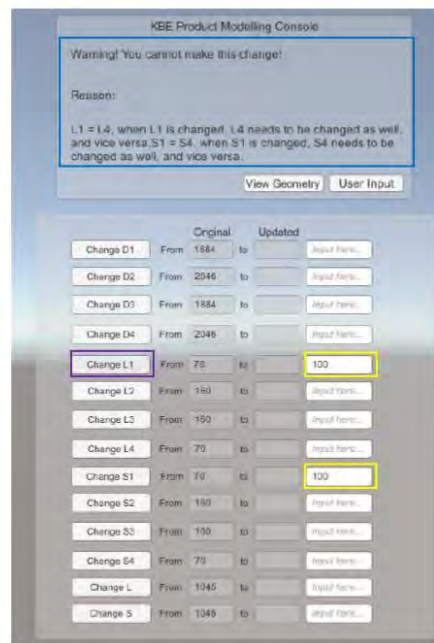


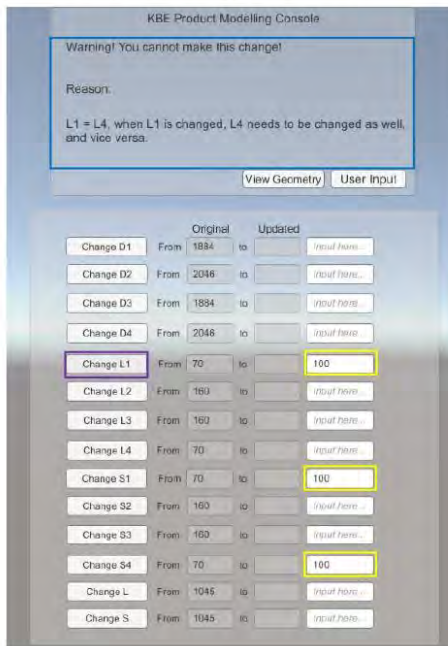
Figure 19: Functions of making possible changes to the wheel assembly model in the developed knowledge-based product modelling environment (*print in colour*)



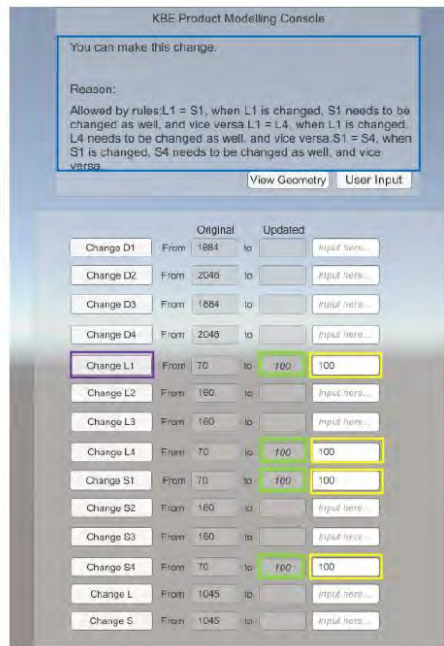
(a) input L1 and apply the change



(b) input L1, S1 and apply the change



(c) input L1, S1, S4 and apply the change



(d) input L1, S1, S4, L4 and apply the change

Note: Yellow box - user input; Green box - propagated parameter (changes allowed by rules); Purple box – button pressed to apply the change; Blue box - knowledge reasoning.

Figure 20: Results of validation – use case 3: wheel assembly, scenario one - change the wheel part dimension - L1 parameter (*print in colour*)

4. Discussion

In comparison to the existing/legacy CAD systems (see Table 8), the developed knowledge-based product modelling system has shown extended capabilities of knowledge reasoning, reuse, and exchange to enhance the product modelling process. The successful implementation of three use cases has proved the effectiveness of the prototype system in reusing the existing product information as knowledge to provide knowledge reasoning in the product modelling process. The visualisation of product geometry and its associated knowledge has validated the proposed data exchange method for exchanging the geometric data and knowledge (non-geometric data) of a product.

Table 8. Comparison of the use case implementation results between the existing/legacy CAD system and proposed system

Evaluation Criteria	Existing/legacy CAD system implementation	Knowledge-based product modelling prototype system
The capability of visualising the product geometry and its associated knowledge	Visualisation of product geometry and its change. Knowledge visualisation is not available.	Visualisation of the original geometry and associated knowledge. Text visualisation of the changes to the geometry.
The capability of presenting every part of the product and the relationships among them	Representation of assembly relationships in the hierarchy tree.	The assembly is shown as “assembly”, and the parts are shown as “part” in the tool interface.
The capability of propagating changes of parameters to drive and constrain the product geometry by reuse of the existing knowledge	Manual tracking of changes in parameters. The change of geometry is reflected in 3D visualisation graphically. No knowledge reasoning is provided when changing parameters in the existing CAD systems. Rules are not available for	Automatic tracking of changes of parameters through the reuse of rules. Text visualisation of the affected parameters in the tool interface. The change of geometry is reflected through text description due to the limited available enabling tools. Knowledge reasoning is

	reuse.	provided in the interface by the reuse of rules from the existing knowledge.
The correctness of the changes applied to the product geometry by reuse of the existing knowledge.	Manual check-up of the correctness of changes is required.	The changes applied to the geometry follow the rules derived from existing information. The correctness is proved during the validation stage.
The capability and correctness of the product geometry data exchange between different platforms	The geometric data is correctly exchanged in the format of a STEP file.	The geometric data is exchanged in the format of a STEP file. The correctness is proved during the visualisation process.
The capability and correctness of the knowledge exchange through the knowledge file	The existing/legacy CAD systems cannot exchange knowledge with each other using a generalised format.	The knowledge is stored and exchanged in knowledge files in XML. Knowledge exchange is proved during the implementation and validation stage.

From the result of the first use case, it can be seen that there is no significant distinction between modelling simple parts by using the proposed system and using existing/legacy CAD systems due to the simplicity of the product and its associated knowledge. Design rules involved in the modelling process are uncomplicated, and only a few parameters could be used to vary the model. In this case, the effectiveness of knowledge reuse in the product design process through using the proposed system may not be evident because understanding the design rules and modelling these simple parts is straightforward; Users would model the simple parts quickly and correctly even without the support of knowledge reasoning and using the existing product information as guidance.

However, the result of use cases 2 and 3 show that the difference in product modelling between using the prototype system and existing/legacy CAD systems becomes more evident when the product becomes more complex and has more design rules and parameters. Given adequate existing product information, the developed knowledge-based product modelling system could provide sufficient knowledge reasoning to design engineering parts and assembly. The explicit design knowledge could help users such as less-experienced design

engineers, multidisciplinary teams involved in the product development process, and non-engineers to understand the product better and identify the essential data according to their design specifications. The reuse of design rules that define connections and constraints between different internal and external parameters would help generate more product variants with different combinations of parameters in an agile way. The knowledge reasoning and reuse would help users automatically check the correctness of changes according to the captured design rules. This would save the time needed for making changes and eliminate errors during the modelling process. And the knowledge exchange using knowledge files provides a steady data exchange method for storing and transferring the captured knowledge between different knowledge-based product modelling platforms through an interoperable XML format. Therefore, the enhancement to the product design process by using the proposed system will become more evident when modelling products with more complexity regarding existing product information, parameters, internal and external rule constraints and relations.

The implementation and validation results also help recognise the limitation of the developed knowledge-based product modelling prototype system in visualisation. The possible changes of dimensions that the users in this interface can make are limited to the use case. And the results of changes to the product, such as changes to the length, width, height, diameter, and material, are presented through the text description in the interface. The text description is not as effective as a graphical 3D display in visualising geometry. This deficiency is caused by the lack of available enabling technologies that support editing the geometry data in the STEP file and displaying the graphical changes directly in the modelling environment. However, the limitation in visualisation is acceptable as the system has shown its effectiveness in propagating the changes of simple parts geometry by reusing the captured knowledge.

5. Conclusions and future work

This paper proposed a knowledge-based product modelling prototype system developed in a game engine platform (Unity) to enhance engineering product design. This prototype system is developed and implemented based on the Virtual Product Modelling framework introduced by the authors. Design rules were reused to provide knowledge reasoning interactive mechanism to help provide a comprehensive understanding of the product, reduce the time of making changes to different connected parameters and avoid errors in the product design process. Three use cases have been implemented to validate the workability and effectiveness of the proposed framework and prototype system. Critical analysis of the implementation results between product design in the existing/legacy CAD systems and the proposed system proved the advantages of this research and highlighted the capability of knowledge reasoning and reuse of the developed knowledge-based product modelling system. These advantages include visualisation of the product's associated knowledge, automatic tracking of changes in parameters, text visualisation of the changes between the original and modified product models with the affected parameters, knowledge reasoning by reuse of the existing knowledge and knowledge exchange in a platform-independent neutral format for data communication. The resulted enhancement to the product design process by using the proposed system lies in modelling products with more complexity regarding existing product information, parameters, internal and external rule constraints, and relations.

The developed knowledge-based product modelling system is limited in visualising and changing the geometry of the product models. The visualisation function can be further improved through the use of a mature and enabling CAD engine in the future. Moreover, the concept of using rules to provide knowledge reasoning for engineering product design shows another possible direction to expand this research outcome into the current CAD product

modelling legacy systems to help the development of a more functional and powerful knowledge-based product modelling engine.

Since the knowledge would be captured from experienced designers and be accessible to new designers through the implementation of the VPM framework, this system can be considered to support non-engineers who lack the engineering background knowledge to understand and learn product design and to carry out design tasks in a more user-friendly way. The integrated knowledge in the proposed system can also be used as a knowledge repository that provides available and accessible product knowledge for non-engineers to thoroughly understand the purpose of each design detail in the existing CAD models and to unravel the complex design references created by other designers.

Some recent research work offers the potential for the use of Virtual Reality (VR) and Augmented Reality (AR) technologies in the field of engineering design and modelling[31], [32]. With the utilisation of headsets, glasses, controllers and sensors, VR and AR provide an immersive way of visualisation and making interaction with virtual and real environments. Thus, the proposed system can possibly be applied together with VR and AR applications to enable design engineers to visualise the product and perform the product modelling process in new ways.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Reference

- [1] S. Suresh and C. Egbu, 'Key issues for implementing knowledge capture initiatives in small and medium enterprises in the UK construction industry', 2006.
- [2] S. Cooper, I. Fan, and G. Li, 'Achieving Competitive Advantage Through Knowledge-Based Engineering A Best Practice Guide', p. 21, 2001.
- [3] K.-H. Chang, 'Solid Modeling', in *e-Design*, Academic Press, 2015, pp. 125–167.
- [4] F. Arnold and G. Podehl, 'Best of both worlds - A mapping from EXPRESS-G to UML', *Unified Model. Lang. «UML» '98 Beyond Not.*, pp. 49–63, 1999, doi: 10.1007/978-3-540-48480-6_5.
- [5] H. Kahn, N. Filer, A. Williams, and N. Whitaker, 'A generic framework for transforming EXPRESS information models', *CAD Comput. Aided Des.*, vol. 33, no. 7, pp. 501–510, 2001, doi: 10.1016/S0010-4485(01)00049-5.
- [6] R. Barbau *et al.*, 'OntoSTEP: Enriching product model data using ontologies', *Comput. Des.*, vol. 44, pp. 575–590, 2012, doi: 10.1016/j.cad.2012.01.008.
- [7] L. Blessing and K. Wallace, 'Supporting the Knowledge Life-Cycle', in *Proceedings of the IFIP TC5 WG5.2 Third Workshop on Knowledge Intensive CAD*, 1998, pp. 21–38, doi: 10.1007/978-0-387-35582-5.
- [8] G. La Rocca, 'Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design', *Adv. Eng. Informatics*, vol. 26, no. 2, pp. 159–179, 2012, doi: 10.1016/j.aei.2012.02.002.
- [9] C. Chapman, S. Preston, M. Pinfold, and G. Smith, 'Utilising enterprise knowledge with knowledge-based engineering', *Int. J. Comput. Appl. Technol.*, vol. 28, no. 2–3, pp. 169–179, 2007, doi: 10.1504/IJCAT.2007.013354.
- [10] E. M. Shehab and H. S. Abdalla, 'Manufacturing cost modelling for concurrent product development', *Robot. Comput. Integr. Manuf.*, vol. 17, no. 4, pp. 341–353, 2001, doi: 10.1016/S0736-5845(01)00009-6.
- [11] I. O. Sanya and E. M. Shehab, 'An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry', *Int. J. Prod. Res.*, vol. 53, pp. 1–27, 2014, doi: 10.1080/00207543.2014.965352.
- [12] I.-S. Fan and P. Bermell-Garcia, 'International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management', *Concurr. Eng.*, vol. 16, no. 4, pp. 271–277, 2008, doi: 10.1177/1063293X08100027.
- [13] M. Cederfeldt, F. Elgh, and I. Rask, 'A Transparent Design System for Iterative Product Development', *J. Comput. Inf. Sci. Eng.*, vol. 6, pp. 300–307, 2006.
- [14] R. Curran, W. J. C. Verhagen, and M. J. L. Van Tooren, 'The KNOMAD methodology for integration of multi-disciplinary engineering knowledge within aerospace production', *48th AIAA Aerosp. Sci. Meet. Incl. New Horizons Forum Aerosp. Expo.*, pp. 1–16, 2010.
- [15] A. T. A. Chiang, A. J. C. Trappey, and C. C. Ku, 'Using Knowledge-Based Intelligent Reasoning To Support Dynamic Collaborative Design', in *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference 2004*, 2004,

pp. 1–12.

- [16] G. La Rocca, *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*. 2011.
- [17] O. Isaksson *et al.*, ‘Trends in Product Modelling - an ENDREA perspective’, 2000.
- [18] Melody Stokes, *Managing Engineering Knowledge: MOKA - Methodology for Knowledge Based Engineering Applications*. Wiley-Blackwell, 2001.
- [19] M. Barnes, ‘Introduction to Collada’, 2007.
<https://www.gamedeveloper.com/art/introduction-to-collada> (accessed Dec. 06, 2021).
- [20] Khronos Group, ‘COLLADA Overview’, 2004. <https://www.khronos.org/api/collada> (accessed Dec. 06, 2021).
- [21] ISO/PAS 17506, ‘Industrial automation systems and integration — COLLADA digital asset schema specification for 3D visualization of industrial data’, 2012.
<https://www.iso.org/standard/59902.html> (accessed Dec. 06, 2021).
- [22] J. Haas, ‘A History of the Unity Game Engine for An Interactive Qualifying Project’, p. 44, 2014.
- [23] S. K. Arora, ‘Unity vs Unreal Engine: Which Game Engine Should You Choose?’, 2021. <https://hackr.io/blog/unity-vs-unreal-engine> (accessed Dec. 06, 2021).
- [24] Unity Technologies, ‘Unity solutions for architecture, engineering and construction’, 2021. <https://unity.com/solutions/architecture-engineering-construction>.
- [25] A. Juliani *et al.*, ‘Unity: A General Platform for Intelligent Agents’, no. February, 2018, [Online]. Available: <http://arxiv.org/abs/1809.02627>.
- [26] A. Hussain, H. Shakeel, F. Hussain, N. Uddin, and T. L. Ghouri, ‘Unity Game Development Engine : A Technical Survey’, *Univ. Sindh J. Inf. Commun. Technol.*, vol. 4, no. 2, pp. 73–81, 2020.
- [27] G. Zhong, V. C. Vijay, and I. Oraifige, ‘A Game-Based Product Modelling Environment for Non-Engineer’, *ICSGGBL 2021 23rd Int. Conf. Serious Games Game-Based Learn.*, vol. 15, no. 4, pp. 308–315, 2021.
- [28] Unity Technologies, ‘Unity User Manual’, *Unity Documentation*, 2020.
<https://docs.unity3d.com/Manual/index.html> (accessed Jun. 18, 2020).
- [29] M. C. Leu, ‘NX10 FOR ENGINEERING DESIGN’, *Design*. p. 207, 2016.
- [30] Object Management Group, ‘OMG Unified Modeling Language’, *Informatik-Spektrum*. 2015, doi: 10.1007/s002870050092.
- [31] O. Huerta, E. Unver, R. Aslan, A. Kus, and D. Chotrov, ‘Application of VR and AR Tools for Technical Drawing Education’, no. May, pp. 363–366, 2019, doi: 10.14733/cadconf.2019.363-366.
- [32] N. Memarsadeghi and A. Varshney, ‘Virtual and Augmented Reality Applications in Science and Engineering’, *Comput. Sci. Eng.*, vol. 22, no. 3, pp. 4–6, 2020, doi: 10.1109/MCSE.2020.2987151.