

University of Southampton Research Repository

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

Thesis: Author (Year of Submission) "Full thesis title", University of Southampton, name of the University Faculty or School or Department, PhD Thesis, pagination.

Data: Author (Year) Title. URI [dataset]

UNIVERSITY OF SOUTHAMPTON

THE FACULTY OF ENGINEERING AND THE ENVIRONMENT

Computational Engineering and Design Research Group

**Automated Design Knowledge Capture as an Aid for Improved
Decision Making and Product Cost Reduction Activities**

by

Vytautas Moncys

*A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Engineering*

September 3, 2019

Supervisors: Prof. James Scanlan
Dr. Murat Hakki Eres
Dr. Stuart Jinks

Declaration of Authorship

I, Vytautas Moncys, declare that this thesis titled, “*Automated Design Knowledge Capture as an Aid for Improved Decision Making and Product Cost Reduction Activities*” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

Vytautas Moncys

Automated Design Knowledge Capture as an Aid for Improved Decision Making and Product Cost Reduction Activities

This thesis presents the development and evaluation of the automated design knowledge capture methodology that aids design decisions and cost reduction activities.

While a number of methods to capture and represent the design knowledge and also methods to aid the decision making are addressed by existing research, all of them focus on separate product development domains or partially linked domains. Furthermore, the existing methodologies require human intervention in one way or another that interrupts the engineers' work-flow. Therefore, this research addresses the mentioned gaps by establishing the relational links between the product development domains and capturing the design knowledge in an automated way without interrupting the work-flow.

The proposed methodology is based on the network interdependency model where links between the nodes serve as information carriers allowing the data to be fed in a forward or backward direction. The links between the requirements, design and cost domains capture the relational dependency information as well as the domain data which is then used for calculation and visualisation purposes. Better understanding of the relational complexities and interactions between the domains leads to better decision making. The framework follows the work-flow of the product development life cycle and captures targeted design knowledge in an automated way.

The three case studies, varying in size and complexity, demonstrate a successful application of the methodology. In addition, the calculated requirements impact highlights the requirements that are linked to the high cost features as well as the number of interdependencies between the requirements. The design knowledge map helps to quickly identify the potential cost reduction areas and highlights which design features and requirements might be affected by any changes in the system. Having a holistic view of the relational dependencies between the requirements, design and cost domains together with the associated data enables the cost reduction scenarios to be simulated in a shorter time resulting in the significantly reduced overall cost and trade off analysis time. The evidence from the second and third case studies suggests that the same level of effort is required applying the framework on one component and on the mini sub-system of three components. However, more empirical studies are necessary to confirm that the proposed methodology is scalable when applied to the large sub-systems, such as IP Turbine.

Acknowledgements

First and foremost I would like to thank my academic supervisors who kept me on track throughout my EngD program at the University of Southampton. I want to thank Prof Jim Scanlan for his guidance and advice that helped me stay focused and his support during the difficult as well as the breakthrough moments; Dr Hakki Eres for his encouragement and his insightful advice.

Besides my academic supervisors, my research work was also supported by industrial supervisors at Rolls-Royce. Therefore, I would like to say big thank you to Dr Phani Chinchapatnam for his effort to introduce me to the right people in the company and to Dr Stuart Jinks for his pragmatism. I will always be grateful for this opportunity to Dr Steve Wiseall who sadly passed away during the first year of my EngD.

My thanks must also go to my colleagues in the Cost and Value Engineering team at Rolls-Royce for sharing their knowledge and expertise, in particular, to Dr Mani Seethapathy, for his coding lessons and Dr Christopher Dodd for proofreading my thesis. Special thanks must go to the BOXARR team who invested significant time and effort in supporting this research. I need to thank my fellow EngDs and friends for all the humour and fun times we had together during my research.

I would also like to mention that this work was funded by Rolls-Royce plc under SILOET II programme and Engineering and Physical Sciences Research Council (EPSRC). Because without funding this project would never have happened.

Last but not least, I would like to thank my parents, without whom I would not be the person I am today; my little brother for his support; my wife Renata who always encouraged and supported me throughout the course of this EngD; and my son Domantas who was born during the final year of my Engineering Doctorate degree and who is a source of motivation.

Conferences and publications

Presented at the Engineering Doctorate Conference 2015. Solving Global Engineering Challenges. University of Southampton, November 2015.

Presented at the Rolls-Royce UTC Conference 2017. Rolls-Royce plc, October 2017.

A journal publication is being prepared.

Contents

Declaration of Authorship	iii
Abstract	v
Acknowledgements	vii
Conferences and publications	ix
Contents	xi
List of Figures	xv
List of Tables	xix
Nomenclature	xxi
1 Introduction	1
1.1 Introduction	1
1.2 Motivation	2
1.3 Research Objectives	3
1.4 Research Scope	4
1.5 Outline of the Thesis	4
2 Literature Survey	7
Knowledge Management and Design Rationale	7
2.1 Knowledge Management	7
2.2 Knowledge in Product Design	9
2.3 Knowledge Based Engineering (KBE) Systems	12
Summary	14
2.4 Design Rationale	14
2.5 Design Rationale Capture and Editing Tools	15
2.5.1 Argumentation-based Design Rationale	16
2.5.1.1 Issues-Based Information System	16
2.5.1.2 Procedural Hierarchy of Issues	16
2.5.1.3 Decision Representation Language	17
2.5.1.4 Questions, Options, and Criteria	18
2.5.2 Derivative Design Rationale Systems	19
2.5.3 Graphical Design Rationale Editors	19
2.5.3.1 Compendium	20
2.5.3.2 Design Rationale Editor	21
2.5.4 Action-based and Model-based Design Rationale	22
2.5.4.1 Design History Tool	22
2.5.4.2 Grammar and Extended Dynamic Programming	23

2.5.4.3	Rationale Construction Framework	26
2.6	Design Rationale Capture During Detailed Design Stage	27
2.7	Use of Annotations in Mechanical Design Context	28
	Summary	30
	Geometry Representation and Modelling	30
2.8	Mechanical Engineering Design Support Systems	30
2.9	The Specific and Generic Systems	31
2.10	Geometry Representation	32
2.11	Geometry Modelling	32
	Summary	34
	Requirements Engineering	35
2.12	Requirements Prioritisation	35
2.12.1	Analytical Hierarchy Process	35
2.12.2	Planning Game	36
2.12.3	Binary Search Tree	36
2.12.4	Cumulative Voting	37
2.12.5	Cost-Value Approach	37
2.13	Requirements Interdependency	37
2.14	Requirements Traceability	38
	Summary	38
	Cost Estimation	39
2.15	Cost Estimation Techniques	39
2.15.1	Parametric	40
2.15.2	Case Based Reasoning	40
2.15.3	Bottom-Up	40
2.15.4	Expert Judgement	41
2.15.5	Feature Based Costing	41
2.15.6	Activity Based Costing	42
	Summary	42
	Chapter Summary	42
3	Critical Literature Review and Research Focus	45
3.1	Challenges in Design Rationale Capture	45
3.1.1	Prescriptive and Descriptive Approach to Design Rationale Capture	45
3.1.2	Advantages and Disadvantages of Prescriptive and Descriptive Approaches	46
3.1.3	Characteristics of Prescriptive and Descriptive Approaches	47
3.1.3.1	When	47
3.1.3.2	What	47
3.1.3.3	Whose	48
3.1.3.4	Who	48
3.1.3.5	From Where	48
3.1.4	Prescriptive or Descriptive	49
3.1.5	The Dilemma of Knowledge Capture Tools	50
3.2	What Knowledge Resides in Product?	51
3.2.1	The Case Study	52
3.3	Challenges in Requirements Engineering	61

3.4	Research Focus	64
	Chapter Summary	65
4	Development of the Decision Support Methodology	67
4.1	Introduction	67
4.2	Requirements Document	68
4.3	Clustering of the Requirements in BOXARR	72
4.3.1	BOXARR Software	72
4.3.2	Clustering of the Requirements	73
4.4	Key Driving Requirements	74
4.5	Design Features	80
4.6	Vanguard Cost Models	80
4.7	The Methodology	85
5	Example Test Case: Rear Lock Plate	95
5.1	Rear Lock Plate	95
5.2	Methodology Demonstration	97
5.2.1	The Result	109
5.3	Methodology Demonstration: The Rear Lock Plate Redesign Activity	117
5.4	Discussion	123
5.5	Lessons Learned	123
6	Example Test Cases: IPT Stub Shaft and Triple Seal Flange Joint	125
6.1	Methodology Improvements	125
6.1.1	Automated Design Features Extraction from CAD	125
6.1.2	Semi-automated Tagging of the Design Features in Vanguard Cost Model	127
6.1.3	Automated Link Creation between the Requirements and Design Features Using Data from the DFMEA Tool	128
6.1.4	Material Cost of the Feature	132
6.2	IPT Stub Shaft Case Study	133
6.2.1	The Set-up	133
6.2.2	The Case Study	133
6.2.2.1	Requirements Domain	134
6.2.2.2	Design Domain	134
6.2.2.3	Cost Domain	135
6.2.2.4	BOXARR	137
6.2.3	Result and Discussion	137
6.3	Triple Seal Flange Joint Case Study	145
6.3.1	The Case Study	145
6.3.2	Result and Discussion	146
7	Conclusions and Future Work	157
7.1	Conclusions	157
7.2	Future Work	162
	Appendices	163

Appendix A	163
Appendix B	168
Appendix C	171
Appendix D	176
Appendix E	179
Appendix F	184
Appendix G	190
References	192

List of Figures

1.1	Research scope.	5
2.1	Results from the study carried out by Maksimovic et al. (2014).	8
2.2	Knowledge Classification. Adapted from Chandrasegaran et al. (2013) . .	10
2.3	Five categories of knowledge representation. Adapted from Chandrasegaran et al. (2013).	11
2.4	Knowledge representations in product design (Chandrasegaran et al., 2013). Here, VOC – Voice Of the Customer; GD&T – Geometric Dimensioning and Tolerancing.	12
2.5	Product Life Cycle Cost. Adapted from Verhagen et al. (2012).	13
2.6	Prediction of Product Lifecycle Cost with increased available knowledge in early design stages. Adapted from Verhagen et al. (2012).	13
2.7	gIBIS example. Adapted from Lee and Lai (1992).	17
2.8	PHI model. Adapted from Chan (2007).	18
2.9	DRL model. Adapted from Lee and Lai (1991).	19
2.10	The components of a design space represented by QOC notation. Adapted from MacLean et al. (1993).	20
2.11	Compendium node types and argumentation structure, an example from the Compendium user instructions.	21
2.12	DRed elements and statuses (Aurisicchio and Bracewell, 2013).	23
2.13	DRed example. Diagnosing a problem (Bracewell et al., 2009).	24
2.14	The DHT model. Adapted from Wiegeraad (1999).	25
2.15	Decision-chain (Nagy et al., 1992).	25
2.16	Decision-process (Nagy et al., 1992).	26
2.17	Issue-decomposition (Nagy et al., 1992).	26
2.18	A three layer design process model (Ishino and Jin, 2002).	27
2.19	The hierarchical tree structure of product models: P ₁ - initial product model; P ₉ - final product model that meets the requirements (Ishino and Jin, 2002).	27
2.20	Specific and Generic systems.	31
2.21	Feature Types.	33
3.1	The aspects that characterises the prescriptive approach. Reproduced from Chan (2007).	49
3.2	The aspects that characterises the descriptive approach. Reproduced from Chan (2007).	50
3.3	The knowledge capture tools currently used in the product design process.	50
3.4	The aspects addressed by the descriptive approach and generic design tools. Adapted from Chan (2007).	51
3.5	The aspects addressed by the prescriptive and descriptive approaches to design rationale capture. Adapted from Chan (2007).	51
3.6	The Rear Lock Plate CAD model.	53

3.7	The Rear Lock Plate drawing.	54
3.8	The Rear Lock Plate part model with the defined design features.	54
3.9	Part models with and without the information about the design features.	55
3.10	Design feature ‘ <i>Foot Contact Pads</i>	56
3.11	Design features: <i>Foot Contact Pads</i> , <i>Stiffening Ribs</i> and <i>Weight Saving Pockets</i>	56
3.12	Relational data could be accessed via the Dependency panel in part navigator pane.	57
3.13	Relational dependencies between the modelling features. The design features are defined.	58
3.14	Relational dependencies between the modelling features. The design features are not defined.	59
3.15	Filtered upstream and downstream connections of the ‘ <i>RH OVERLAP CUTTER</i> ’ modelling feature.	60
3.16	Number of documents that have the words ‘ <i>requirements engineering</i> ’ in the title. ¹	62
3.17	The type of documents that have the words ‘ <i>requirements engineering</i> ’ in the title. ¹	62
4.1	The Methodology Framework with indicated automation steps.	69
4.2	The roadmap to create links between Requirements, Design, and Cost domains	69
4.3	The concept of BOXARR. Adapted from (BOXARR, 2016).	73
4.4	Example of Component Requirements Document and its structure. Adapted from (Carter and Saunders, 2015).	75
4.5	Example of listed regulatory framework requirements. Adapted from (Carter and Saunders, 2015).	76
4.6	The comparison of requirements data in Microsoft Word document and in Boxar. Adapted from (Carter and Saunders, 2015).	77
4.7	Clustered requirements in BOXARR.	78
4.8	Example of how imported data looks in BOXARR.	79
4.9	Hierarchical tree structures and functional blocks in Vanguard. Adapted from (Vanguard, 2016).	81
4.10	Cost modelling template used by Rolls-Royce (Rolls–Royce, 2016).	82
4.11	The workflow of Vanguard Studio (Rolls–Royce, 2016).	84
4.12	Product development life-cycle stages.	86
4.13	Importing data into BOXARR.	87
4.14	The <i>Requirements Importance Weight</i> is assigned automatically by BOXARR and displayed in the designated data field (red rectangle).	89
4.15	The unit cost structure in System, Sub-system and Component levels.	90
4.16	Representation of the component template report that characterizes 7 value streams (Rolls–Royce, 2016).	90
4.17	Representation of the manufacturing operation sequences and their cost.	91
5.1	The CAD model of the Rear Lock Plate.	96
5.2	The components that rear lock plate interfaces with.	96
5.3	Design features of the rear lock plate.	97
5.4	The summary of the rear lock plate requirements.	98

5.5	Representative Vanguard Cost Model of the Rear Lock Plate (Rolls–Royce, 2016).	99
5.6	The representative operations required to manufacture the features of the lock plate (Rolls–Royce, 2016).	100
5.7	Component requirements in CSV file format.	101
5.8	Three domains (Requirements domain, Design domain and Cost domain) created in BOXARR.	101
5.9	The comparison of the Design Features domain with and without the pictures.	103
5.10	The filtered rear lock plate requirements in BOXARR.	104
5.11	The links between the design features and the corresponding operation sequences.	105
5.12	Summary of the calculated weight factors.	106
5.13	The impact of different <i>Requirements Importance Factor</i> weight values.	107
5.14	The mapped requirements to design features.	108
5.15	The holistic map of the relations between the requirements, design and cost domains.	109
5.16	The graph structure.	110
5.17	The graph structure.	111
5.18	Design features of the Rear Lock Plate and the links to the OP sequences.	112
5.19	Filtered downstream connections of a particular requirement.	113
5.20	Filtered downstream connections of a particular requirement with the graphical representation of features.	115
5.21	Filtered upstream and downstream connections of a particular feature.	116
5.22	Calculated impact of the requirement.	118
5.23	Summarised impacts of all 16 requirements.	119
5.24	Representative features manufacturing cost of the rear lock plate.	120
5.25	The CAD model of the redesigned rear lock plate.	120
5.26	Requirements impact summary of the new rear lock plate model.	121
5.27	Representative manufacturing cost of each feature of the new rear lock plate model.	121
5.28	The comparison of the representative value streams of both rear lock plates with and without the ‘ <i>Deflectors</i> ’.	122
6.1	Design Feature grouping.	126
6.2	IPT Stub Shaft Design Features. Coloured names represent the stub shaft features - HP3 Air Seal Fins, Flange Holes, Taper Bolt Holes, Dowel Holes, R030 Holes, Mating Face, Seal Ring Mating Face, Air/Air Lab Seal, Rivet Holes, R044 Holes, Oil/Air Lab Seal, Weld Joint.	127
6.3	Design Features group.	128
6.4	Features option list. The red square indicates that <i>Mating Face</i> feature is ground. The OP cost is representative.	129
6.5	Product development lifecycle. The FFMEA is performed at the start of the project in the requirements domain; Initial DFMEA - at Stage 1 (Preliminary concept definition) and Production DFMEA - at the end of Stage 3 (Product realisation). Although it is not shown in the Figure the PFMEA is performed at the beginning of Stage 3 before the start of production.	130

6.6	Screen shot of the Rolls-Royce DFMEA excel tool.	131
6.7	DFMEA of the IPT Stub Shaft with the guide to perform the analysis. . .	131
6.8	Bearing race.	132
6.9	Condition of supply part feature geometry definition and the calculated representative features mass ratio.	133
6.10	IPT Stub Shaft.	134
6.11	Intermediate geometries of the IPT Stub Shaft.	135
6.12	Geometric features of the IPT Stub Shaft forging model.	136
6.13	IPT Stub Shaft Design Features. Coloured names represent the stub shaft features - HP3 Air Seal Fins, Flange Holes, Taper Bolt Holes, Dowel Holes, R030 Holes, Mating Face, Seal Ring Mating Face, Air/Air Lab Seal, Rivet Holes, R044 Holes, Oil/Air Lab Seal, Weld Joint.	136
6.14	Vanguard cost models for different geometries. All costs are representative.	137
6.15	IPT Stub Shaft knowledge map with the links between requirements do- main (green colour), design domain (yellow colour) and cost domain (dark pink colour). The costs are representative.	138
6.16	Representative features manufacturing and material cost of the IPT Stub Shaft.	140
6.17	The amount of material that needs to be removed for each feature in order to get from the IPT Stub Shaft forging geometry to finished part geomtry.	141
6.18	The impact of the IPT Stub Shaft functional requirements.	142
6.19	Filtered requirements (<i>EDNS01000309547-039</i> and <i>EDNS01000309547- 037</i>) links to features.	143
6.20	The number of requirements that are linked to the ‘HP3 Air Seal Fins’ feature.	144
6.21	The summary of the parameters needed to calculate the impact of each functional requirement for the IPT stub shaft.	145
6.22	System boundary of the triple seal flange joint interface.	146
6.23	Design knowledge of the triple seal flange joint interface sub-system. The costs shown are examples and not the real values.	147
6.24	The impact of the triple seal flange joint functional requirements.	149
6.25	The summary of the parameters needed to calculate the impact of each functional requirement for the triple seal flange joint.	150
6.26	The summary of the representative stub shaft, shaft and disc feature costs.	151
6.27	The summary of the parameters for five requirements used to calculate the requirements impact.	152
6.28	Requirements interdependencies through features.	153
6.29	The summary of the parameters for top five high impact requirements. Stub shaft (top table) and triple seal flange joint system (bottom table). .	155

List of Tables

2.1	Prototype design rationale systems.	20
4.1	The importance of weights on modal verbs.	71
4.2	The importance of weights on modal verbs.	87

Nomenclature

ABC	Activity Based Costing
AHP	Analytical Hierarchy Process
BoM	Bill of Material
B-rep	Boundary Representation
BST	Binary Search Tree
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
COS	Condition of Supply
CRD	Component Requirements Document
CSG	Constructive Solid Geometry
CV	Cumulative Voting
DRM	Design Research Methodology
DFMEA	Design Failure Mode and Effect Analysis
DS-I	Descriptive Study I
DS-II	Descriptive Study II
ID	Identification Document
IPT	Intermediate Pressure Turbine
GD&T	Geometric Dimensioning and Tolerancing
KBE	Knowledge Based Engineering
KBS	Knowledge Based Systems
NURBS	Non-Uniform Rational B-Splines
PDM	Product Data Management
PLM	Product Life-Cycle Management
RE	Requirements Engineering
PS	Prescriptive Study
RC	Research Clarification
R&D	Research and Development
ROM	Rough Order of Magnitude
SCU	Supply Chain Unit
WBS	Work Base Structure

Chapter 1

Introduction

1.1 Introduction

Product design is a complex, often ill-defined, and iterative process. It gradually becomes better defined as it progresses through the product development stages where design problems are solved in a multi-stage, iterative, and collaborative process with extensive communication among teams from various departments. [Tong and Sriram \(1992: p. 1\)](#) describes design as “*a process that constructs a description of an artefact, process, or instrument that satisfies a (possibly informal) functional specification, meets certain performance criteria and resource limitations, is realizable, and satisfies criteria such as simplicity, testability, manufacturability, and reusability*”. Product design process demands a wide variety of knowledge sources, such as heuristic knowledge, qualitative knowledge, and quantitative knowledge ([Chandrasegaran et al., 2013](#)).

With the advent of Information Age ([Mendelson and Pillai, 1999](#)) the amount of data that organizations and people have to process increased dramatically. The same trend is also observed in the design industry. The amount of raw data available to the designer is vast. Therefore, effective and efficient knowledge capture and representation tools are critical for easy retrieval and reuse of information. Moreover, effective transfer of knowledge between teams during the collaborative design process allows companies to reduce product development time and stay competitive as a result.

[Cross \(2006: p. 100\)](#) stated that “*our concern in design research has to be the development, articulation and communication of design knowledge*”. He identified three sources of knowledge: people, processes and products. Design knowledge resides in people: in everyone involved in the product development process. Often, this source of knowledge is kept in people’s heads and in their notebooks and is not readily available for others. Design knowledge residing in processes is in the form of drawings, specifications, design definition reports and user manuals that result from product development processes. This form of knowledge was primarily aimed at providing the information required to manufacture, use and maintain the released product, but not to support the development of the next product generation ([Wiegeraad, 1999](#)). Design knowledge residing in products is expressed in the forms, fits and materials which embody design attributes.

Over the last three decades, researchers proposed and developed a number of tools and methodologies to capture, represent and retrieve design rationale throughout the design process (Kunz and Rittel, 1970; MacLean et al., 1989; Lee and Lai, 1991; Conklin and Yakemovic, 1991; McCall, 1991; Chen, 1991; Nagy et al., 1992; Myers et al., 2000; Ishino and Jin, 2002; Aurisicchio and Bracewell, 2013; Van Schaik, 2013). However, only few successful applications of the design rationale system have made it into practical use in industry (Regli et al., 2000). While the majority of the developed tools and methodologies are aimed at capturing and representing design rationale from people and processes, only a few researchers proposed to link the design rationale directly to the product model (Chen, 1991; Wiegeraad, 1999; Chan, 2007; Van Schaik, 2013). However, none of the proposed methodologies for linking the design rationale directly to the product model have been implemented in industry nor have they been integrated in any of the commercial design tools. Therefore, the effectiveness of these methodologies cannot be assessed from the industrial perspective.

1.2 Motivation

The design decisions made during various stages of the design process have a profound impact on the product life cycle cost. Between 60% and 80% of the total project costs are committed during the early stages of new product development (Court et al., 1997; Verhagen et al., 2012). Therefore, effective and efficient product data management is crucial in the product development process as it defines the longevity of the project, which in turn affects the cost. Moreover, in the early stages of design it is essential to provide effective tools to the designer for better-informed decision making and for better exploration of the design space because often in early design stages the knowledge of the product is unclear, incomplete, and difficult to represent (Chandrasegaran et al., 2013). The developing computer technologies and information management software such as Product Data Management (PDM) or Product Lifecycle Management (PLM) enables engineers and companies to communicate on a global scale. However, the communication and means for communication are still a core characteristic of successful design work (Maier et al., 2009; Li et al., 2009; Heisig et al., 2010). Most design projects are dominated by the redesign activities and only a few are aimed at developing totally new products (Lee, 1997; Wiegeraad, 1999; Chan, 2007). The information about the product, such as function, purpose, detailed description of the final product and rationale that helped materialize the final artefact is extremely important for efficient and effective design processes. The information sources that are currently available to the designers are in the form of drawings, models, reports, plans, minutes and correspondence (Heisig et al., 2010). Investigations by other researchers reveal that available information sources do not support effective re-use of design knowledge and information; they capture only fragments of design processes (Wiegeraad, 1999; Aurisicchio et al., 2010; Aurisicchio et al., 2013). The facts from a survey presented by Heisig et al. (2010) confirm that

further research on design rationale capture and knowledge re-use tools is still needed. Out of 137 respondents, of which the majority were Design Engineers and Managers with more than 10 years' professional experience, the most mentioned category for the 'need to retrieve' knowledge and information was 'rationale'. The need was not only to retrieve rationale of the artefact, but also rationale for changes made during the product life, difficulties during manufacturing and lessons learnt (Heisig et al., 2010: p. 508):

- *'The drawings are always there along with details, but a lot of "detective work" is often involved when realising why features are there/material choices for example and hence how they can be improved'* Service Engineer, Aerospace, 1 year professional experience.
- *'Often previous designs were recorded as a mass of detail without clearly stating what was finally used and why'* Design Engineer & Manager, 25 yrs.
- *'History of all changes to a product and the reasons for the changes'* Product Manager & Support Engineer, Engineering, 41 yrs.

Respondents also stressed that drawings/diagrams should contain 'high-level' and 'low-level' information as well as the 'interactions' between the components. Surprisingly only seven respondents' emphasised the 'design for reuse' as a need for retrieval (Heisig et al., 2010: p. 509):

- *'Where possible, it would be important not to recreate work, so I would seek out pieces of functionality suitable for reuse'* Design Engineer, 1st year.
- *'There needs to be a clear wrap-up of the design process to document the detail behind the one solution that was arrived at specifically with a view to making it easy to reuse later'* Design Engineer & Manager, 25 yrs.
- *'Performance measures of previous designs so that elements of the design can be reused in new design work e.g. an existing specific sub-assembly may be able to be carried into a new design if its performance is well understood.'* Manager, 9 yrs.

The responses suggest that there is a lack of understanding of potential efficiency gains from knowledge re-use in industry. This could also be a potential reason for slow integration of knowledge capture tools in industry (Lee, 1997; Regli et al., 2000; Ishino and Jin, 2002; Chan, 2007; Conway et al., 2008; Aurisicchio and Bracewell, 2013; Van Schaik, 2013).

1.3 Research Objectives

The aim of this research is to assist the design process by mapping the requirements to design features and to cost and aiding decision making as a result.

There are three primary questions-objectives of this research:

1. *Is it possible to create a knowledge map that captures the design process knowledge through the dependency links between the requirements domain, design domain, manufacturing domain and cost domain?*
2. *Is it possible to automate the knowledge capture process? If yes, to what degree can the knowledge capture be automated?*
3. *Will this knowledge map be useful for the design community and in what way?*

The first objective is concerned with developing a holistic map of the interrelationships between design, manufacturing and requirements management which all contribute to the total cost of the product. It is believed that constructing traceable links between requirements, design features and cost will enable engineers to quantify how requirements drive cost. Ultimately understanding and visualising these interrelationships will facilitate better decision making.

The second objective investigates how and to what degree the capture of the design knowledge can be automated. The automation will increase the productivity of the design process and the efficiency of the knowledge capture saving valuable engineers time as a result.

The third objective explores the ‘*pros*’ and ‘*cons*’ of the proposed methodology through the case studies and feedback from the engineers at Rolls-Royce.

1.4 Research Scope

There are five general areas, shown in Figure 1.1, that cover the scope of this research: requirements engineering, design, manufacturing and cost and knowledge management, which is the major focus of this research. Requirements engineering will look into the requirements interdependency and traceability aspect. Design features and how they are utilised in the design and manufacturing domains will be investigated. The manufacturing domain will be integrated within the cost domain due to the fact that all necessary manufacturing information can be retrieved from the cost model. Therefore, it will not be explored further in this research. But for completeness it is included in Figure 1.1. Finally, everything will be brought together through the knowledge management prism. The proposed methodology and tool will be applied to two different aerospace components, which are the rear lock plate and the IPT stub shaft, and to the system of components that represent the triple seal flange joint.

1.5 Outline of the Thesis

The thesis is structured in the following way:

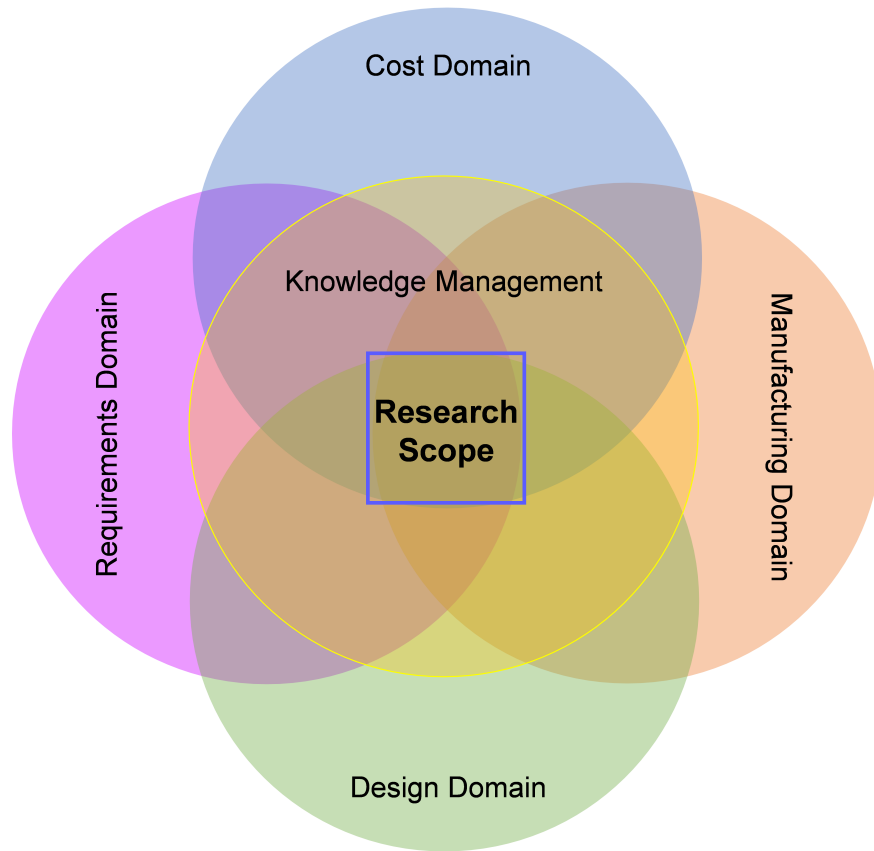


Figure 1.1: Research scope.

Chapter 2 is dedicated to literature review. An overview of knowledge management and knowledge in product design is presented, definitions of design rationale are given and the most notable design rationale capture tools are described. This chapter also explores the design rationale capture during the detailed design stage, discusses the principles of geometry representation and introduces the geometry modelling methods. An overview of requirements engineering is also presented, and requirement prioritisation methods are described. The chapter ends with the introduction of the core techniques and applications in the field of cost estimation, providing the reader with a sufficient understanding of the cost domain.

Chapter 3 presents the critical literature review and identifies the challenges in the fields that have been discussed in Chapter 2.

In Chapter 4, the framework is outlined and the proposed methodology is described. In addition, an overview of each element of the framework is provided.

Chapter 5 demonstrates the application of the proposed methodology using the rear lock plate as an example test case. Two scenarios are presented. First, the design knowledge of the rear lock plate is captured. Second, the captured design knowledge is used in the rear lock plate redesign exercise to assist the decision making.

The methodology improvements are described in Chapter 6 and two case studies presented. The first case study showcases the scalability of the methodology and demonstrates that the *requirements impact* metric is able to highlight the requirements that are linked to the high cost features as well as the interdependencies between the requirements. The second case study demonstrates how the value of the *requirements impact* changes when the same set of requirements is applied to single component and multiple component systems.

Chapter 7 presents the research findings and drawn conclusions, and suggests potential directions for future research.

Chapter 2

Literature Survey

The following chapter presents a foundation for the research in the form of existing literature in the relevant areas that have been categorised into four sections: knowledge management and design rationale, geometry representation and modelling, requirements engineering and cost estimation.

The first section, knowledge management and design rationale, gives an overview of product knowledge and how it is classified in the field of engineering and defines the Knowledge Based Engineering systems. Furthermore, it presents the variety of design rationale capture tools and explores their use cases at different stages of the design process.

The second, geometry representation and modelling, discusses the principles of geometry representation, explores the geometry modelling methods and mechanical engineering design support systems.

The third, requirements engineering, presents a concept of requirements engineering and gives an overview of the requirements prioritisation methods.

The aim of the last section is to introduce the core techniques and applications in the field of cost estimation, providing the reader with a sufficient understanding of the cost domain which is one of the elements of the proposed methodology in this research.

The chapter ends with a summary of the main findings from the four sections. The critical review of the literature is presented in the following chapter.

Knowledge Management and Design Rationale

2.1 Knowledge Management

In the age of information and increased worldwide competition, many engineering enterprises are forced to look for new business development strategies in order to maintain competitive advantage. The full exploitation of a company's intellectual assets

(McMahon et al., 2004; Sainter et al., 2000), in other words, knowledge is critical to competitiveness. According to Ammar-Khodja and Bernard (2008: p. 3), “*Knowledge is regarded nowadays as a strategic resource and a factor of stability, bringing a decisive competing advantage.*” Therefore, knowledge management plays a crucial role in developing new growth strategies for engineering companies in order to gain a competitive edge in the international market. Sainter et al. (2000) emphasises the importance of knowledge management and how it affects a company’s financial resources as well as time spent developing new products.

As technologies advance the products become more complex leading to more people being involved in product development. As a result, more knowledge is generated. Some knowledge could potentially be lost if not managed properly. Without a doubt the knowledge management problem is a complex one (Ammar-Khodja and Bernard, 2008). Over the last two decades academics and people from industry have been trying to tackle the knowledge management problem through research. However, after two decades of intensive research on knowledge management, researchers agree that challenges still remain in this area. The recent work carried out by Maksimovic et al. (2014), investigated the industrial challenges in managing product development knowledge from the perspective of designers and engineers. Participants of the study were from nine different companies operating in five sectors and consisted of 42 designers and engineers with an average of 10-20 years of experience. The research revealed that 57% of the concerns raised in industry were related to knowledge lifecycle activities. Industry experts that participated in the study were mostly concerned about knowledge capture, knowledge sharing, knowledge use and knowledge provision. Challenging areas for each category are summarised in Figure 2.1.

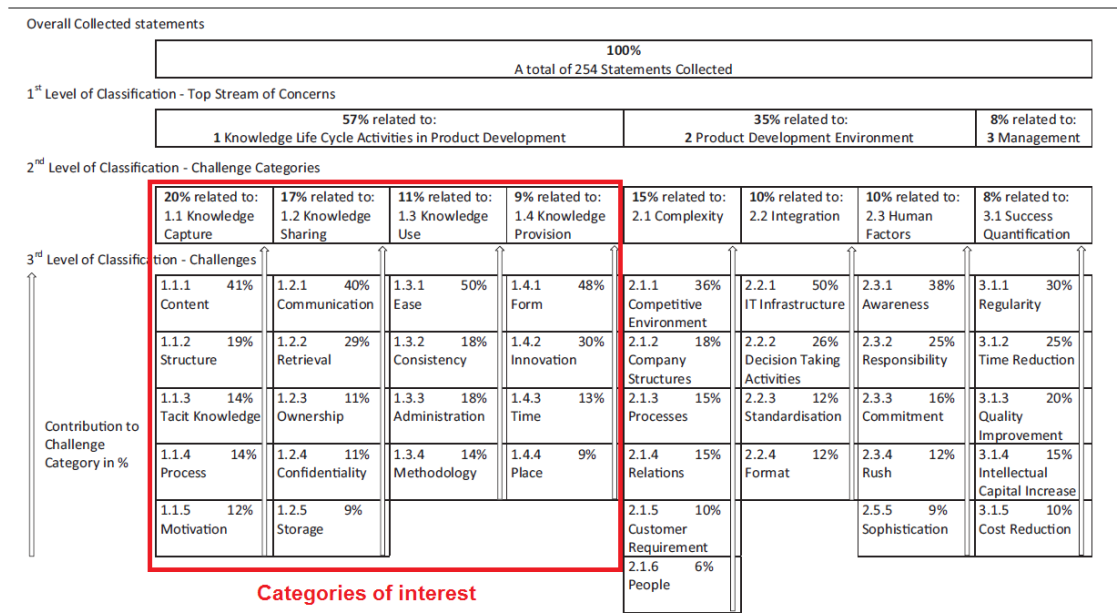


Figure 2.1: Results from the study carried out by Maksimovic et al. (2014).

It is intriguing to note that the most concerning category was the knowledge capture (20%), whereas the areas of underlying challenges were the ease of knowledge use (50%), knowledge content (41%), knowledge communication (40%) and retrieval (29%). This study confirms that more work is needed to be done in the area of knowledge management. In this thesis the Author contributes towards improvement of knowledge communication and retrieval by linking the requirements domain with design, manufacturing, and cost domains and by integrating this process into a Product Development Lifecycle for easier knowledge retrieval.

The next section gives an overview of product knowledge and how it is classified in the field of engineering.

2.2 Knowledge in Product Design

[Chandrasegaran et al. \(2013\)](#) argues that the definition of knowledge, even from the product design perspective alone, depends upon the context and can be defined differently by similar design teams of different products, or by different design teams for the same product. Nevertheless, as long as knowledge is defined, the classification of knowledge can follow. The classification of knowledge in the field of engineering is shown in Figure (2.2).

Formal Vs. Tacit. Knowledge, which is embedded in product documents, repositories, product function and structure description, problem solving routines, technical and management systems, computer algorithms, and expert knowledge systems is classified as formal knowledge ([Owen and Horvath, 2002](#)). Formal knowledge is necessary to build and manufacture a product. Conversely, knowledge related to experiences, intuition, unarticulated models or implicit rules of thumb is termed tacit and is necessary to create new value in a product ([Chandrasegaran et al., 2013](#)). Tacit knowledge resides in people. The knowledge transfer often depends on the willingness of people to share their experiences. Moreover, this knowledge might be lost if people decide to leave the organisation.

Product Vs. Process. Knowledge related to the evolution of a product throughout its life cycle is classified as product knowledge. This includes requirements, various kinds of relationships between parts and assemblies, geometry, functions, behaviour, various constraints associated with products, and design rationale ([Chandrasegaran et al., 2013](#)). Process knowledge can be classified into design process knowledge, manufacturing process knowledge, and business process knowledge.

Compiled Vs. Dynamic. Compiled knowledge can be defined as knowledge of experience that is compiled into rules, plans or scripts, standards, cases of previously solved problems, etc. Dynamic knowledge encompasses qualitative knowledge (commonsense reasoning, approximate theories, causal models of processes, general problem solving knowledge, etc.) and quantitative knowledge (constitutive, compatibility and equilibrium equations, numerical techniques, closed form equations, etc.) ([Sriram, 1997](#)). The solutions in *compiled* and *dynamic* knowledge are *explicit* and *implicit*, respectively

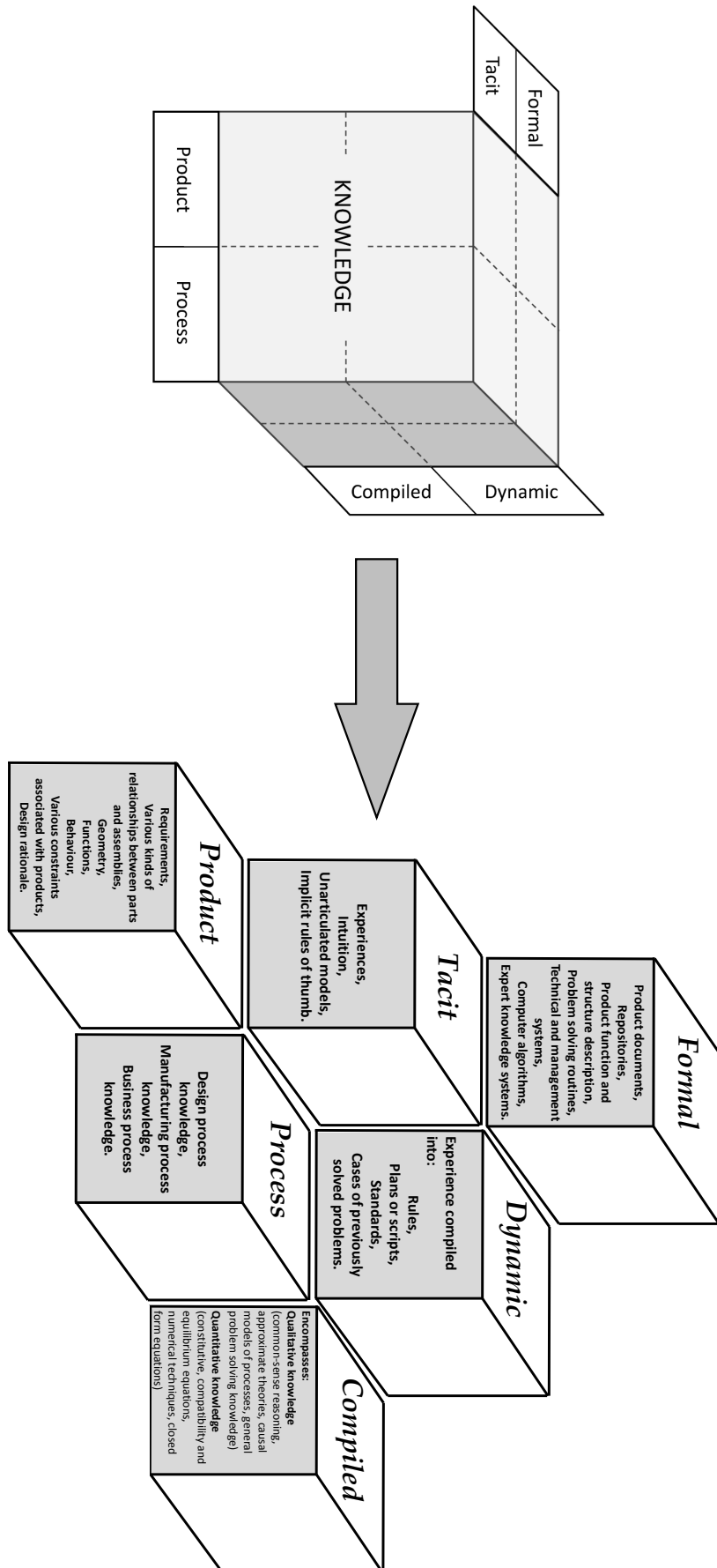


Figure 2.2: Knowledge Classification. Adapted from [Chandrasegaran et al. \(2013\)](#)

(Chandrasegaran et al., 2013). By making knowledge explicit, the ways to represent it can be determined. Owen and Horvath (2002) classifies knowledge representations into five categories: pictorial, symbolic, linguistic, virtual, and algorithmic. Figure (2.3) shows knowledge representations for both product and process knowledge.

Pictorial	Symbolic	Linguistic	Virtual	Algorithmic
Sketches Detailed Drawings Charts Photographs CAD Model Views	Decision Tables Production Rules Flow Charts Assembly Tree Ontologies	Customers Requirements Design Rules Constraints Analogies Customer Feedback Verbal Communication	CAD Models CAE Simulation Virtual Reality Simulations Virtual Prototypes Animations Multimedia	Mathematical Equations Parameterization Constraint Solvers Computer Algorithms Design/Operational Procedures

Figure 2.3: Five categories of knowledge representation. Adapted from Chandrasegaran et al. (2013).

While there have been well-established forms of representing mathematically and geometrically related as well as experimental knowledge, representation of tacit heuristic knowledge is still a developing area. The development of heuristic methods based on geometry attributes, composition, and inheritance for determining mapped concepts in engineering ontologies will lead to the ability to create, share, and exchange knowledge for solving design evaluation problems (Zhan et al., 2010). Moreover, Chandrasegaran et al. (2013) argues that a good product design support tool should not only capture knowledge through the design process, but also represent it on the basis of the relevant context. A significant part of research in product design is focused on the development of tools that support knowledge capture, representation, and reuse throughout the product life-cycle. Figure (2.4) illustrates the concept of knowledge representations in different stages of product design.

It is seen that the amount of information increases with every design stage. It is also evident that knowledge representation in the early stages of design is predominantly linguistic and pictorial in nature. Symbolic, virtual, and algorithmic representations of knowledge appear in the later design stages when much of the design is already committed. As indicated in Figure (2.5) only approximately 25% of validated design knowledge is available by the end of the preliminary design stage, while committed cost accounts for more than 80% of the total product life-cycle cost (Demian and Fruchter, 2009; Verhagen et al., 2012).

It is believed that re-using knowledge will increase the preliminary design knowledge, reduce or even eliminate the duplication of data across systems and reduce the product life-cycle cost (Figure 2.6). While benefits of the tool that supports knowledge capture, representation, and reuse are evident, the development of such a tool seems to remain a challenge.

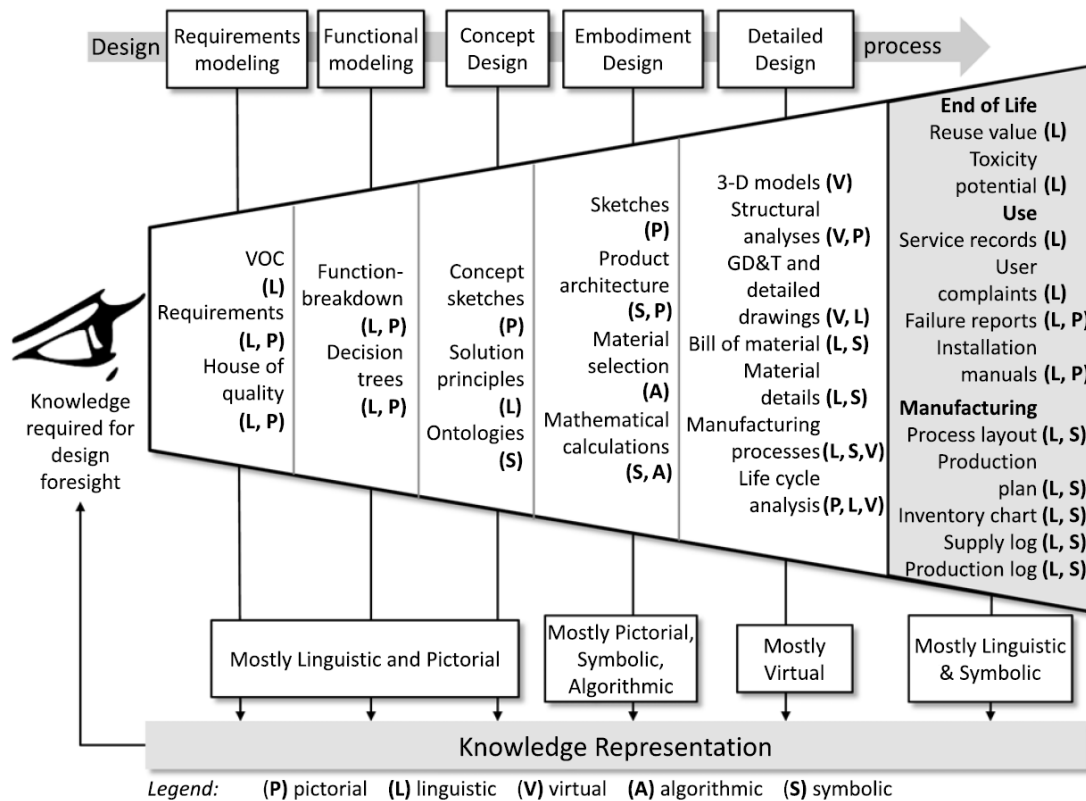


Figure 2.4: Knowledge representations in product design (Chandrasegaran et al., 2013). Here, VOC – Voice Of the Customer; GD&T – Geometric Dimensioning and Tolerancing.

2.3 Knowledge Based Engineering (KBE) Systems

The systems extensively used in the engineering design and manufacturing industry are the Knowledge Based Engineering (KBE) systems. The comprehensive definition of KBE systems is given by La Rocca (2011: p. 57), "Knowledge based engineering (KBE) is a technology based on dedicated software tools called KBE systems, that are able to capture and reuse product and process engineering knowledge. The main objective of KBE is the reduction of time and costs of product development by automating the repetitive, non-creative, design tasks and by supporting the multidisciplinary design optimization in all the phases of the design process." The KBE systems very much succeeded in automating the repetitive design tasks and processes (Cooper et al., 1999; Perry and Ammar-Khodja, 2010). However, the knowledge captured by these systems is not the 'know-why' knowledge, but rather the 'know-how' knowledge. The captured knowledge is encoded into rules that are used by the software to automate the specific design task or process. The KBE systems provide time and cost saving benefits and enhance productivity. However, the main bottlenecks of these systems are that they often operate in a specific expert domain, and require significant development resources and capable

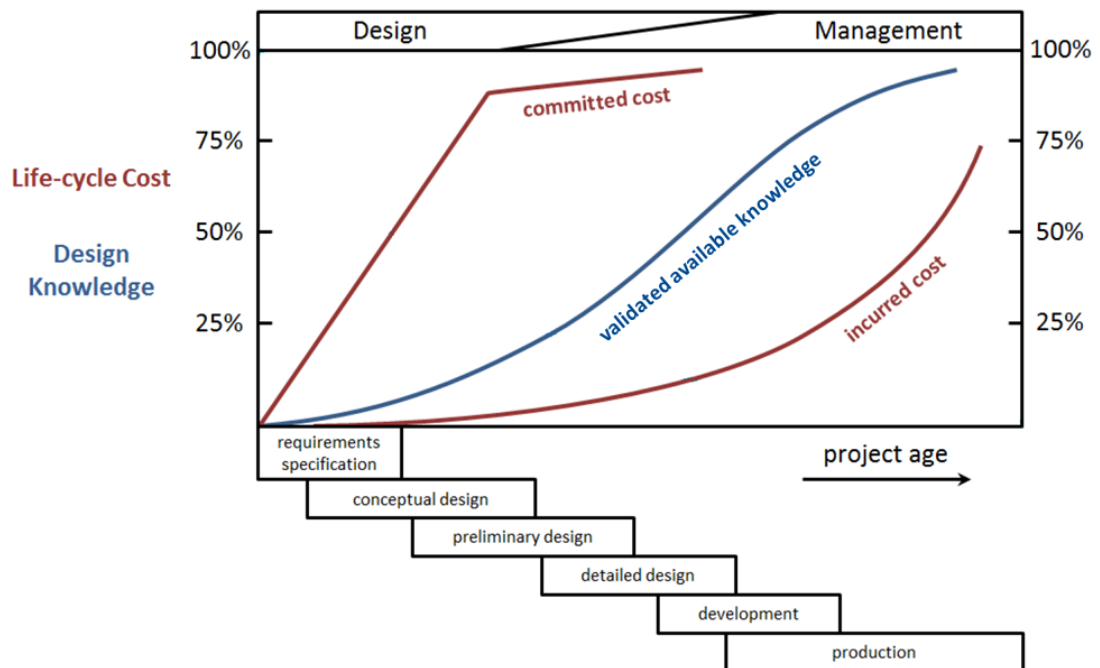


Figure 2.5: Product Life Cycle Cost. Adapted from Verhagen et al. (2012).

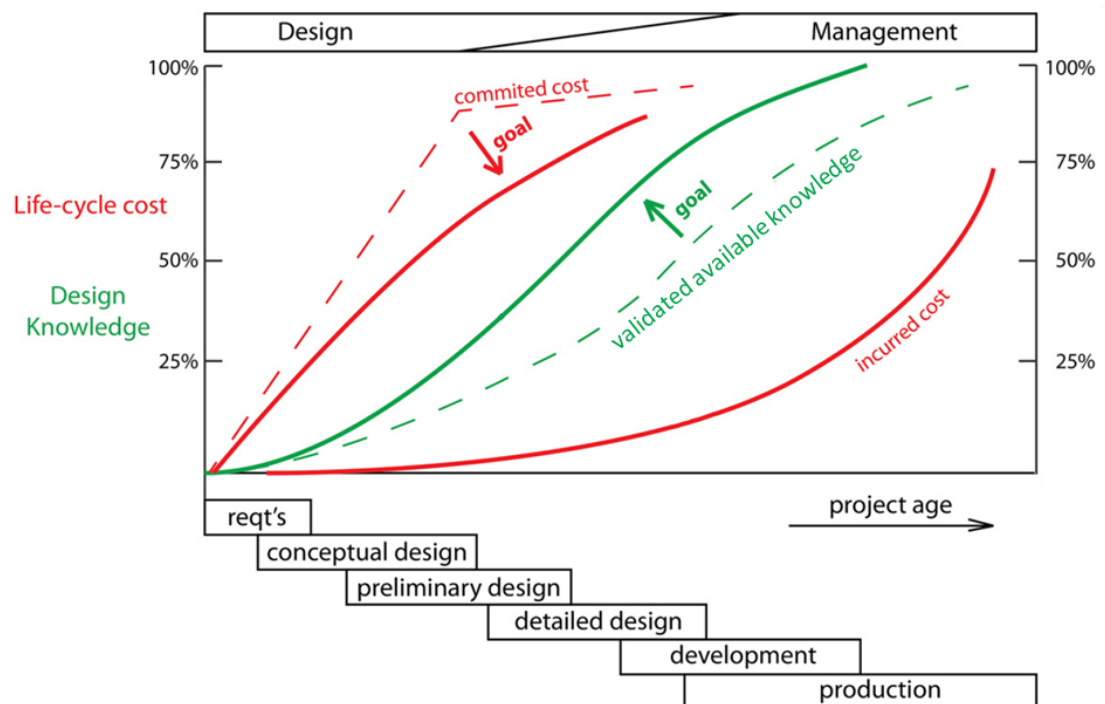


Figure 2.6: Prediction of Product Lifecycle Cost with increased available knowledge in early design stages. Adapted from Verhagen et al. (2012).

application developers who are able to translate the knowledge into computer code and dedicate the resources for maintenance of the software tools (Laan, 2008). It is clear that KBE systems provide great benefits to the designers and engineers when tackling specific repetitive design tasks, however they fail to capture the explicit knowledge on 'why' the product is designed the way it is.

Summary

This section has introduced the KBE systems and the classification of knowledge in product design. The KBE systems provide great benefits to the designers and engineers when tackling specific repetitive design tasks, however they fail to capture the explicit knowledge on 'why' the product is designed the way it is.

Knowledge management plays a crucial role in developing new growth strategies for engineering companies in order to gain a competitive edge in the international market. The example presented in section 2.1 demonstrates how the loss of knowledge or badly written requirements can increase the expenses even when the intention was to reduce the costs.

The amount of information increases with every design stage. Design knowledge representation in the early stages of design is predominantly linguistic and pictorial in nature. Symbolic, virtual, and algorithmic representations of knowledge appear in the later design stages when much of the design is already committed. Only approximately 25% of validated design knowledge is available by the end of the preliminary design stage, while committed cost accounts for more than 80% of the total product life-cycle.

The argument has been put forward that re-using knowledge will increase the preliminary design knowledge, reduce or even eliminate the duplication of data across systems and reduce the product life-cycle cost as a result.

2.4 Design Rationale

Mechanical engineering design relies heavily on past designs or on the use of past experience (Lee, 1997; Kim et al., 2007). Therefore, the information about the previous product, such as function, purpose, detailed description of the final product and rationale that helped materialize the final artefact, is extremely important for efficient and effective new product development. Design rationale encompasses the broader context of the product development process as well as contributing to the evolution of product knowledge base (Szykman et al., 2001).

The term Design Rationale (DR) can be traced back to the pioneering work of Kunz and Rittel (1970) in 1970, addressing ill-structured problems using proposed Issue-Based Information System (IBIS). Originally, the concept was intended to support political and social decision processes when solving complex, 'wicked', problems (Rittel and Webber,

1973). Design rationale system based on IBIS concept gained interest within the research community. Since then, many papers have been published suggesting methods for design rationale capture. The following are definitions of Design Rationale found in literature:

"Design rationale includes not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the trade-offs evaluated, and the argumentation that led to the decision." (Lee, 1997: p. 78)

"Design rationale often means the historical record of the analysis that led the choice of the particular artifact of a feature in question." (Lee and Lai, 1991: p. 256)

"A design rationale is a representation for explicitly documenting the reasoning and argumentation that make sense of a specific artefact." (MacLean et al., 1991: p. 203)

"Design rationale (DR) is the reasoning that goes into determining the design of the artefact. It can include not only direct discussion of artefact properties but also any other reasoning influencing design of the artefact." (Dutoit et al., 2006: p. 3)

"Design Rationale: an information structure that justifies how the implementation (consequences of the design selections) satisfies its specification." (Baxter, 1991: p. 13)

It is clear that the essence of these definitions is the same, only the wording is different. The definition of Design Rationale, formulated by Dutoit et al. (2006), is chosen as a benchmark in this work, since it captures the broader meaning of the term. The fact that DR should *"include not only direct discussion of artefact properties but also any other reasoning influencing design of the artefact"*, strongly coincides with the views of the Author.

In the product development process two main DR goals can be identified (Szykman et al., 2001). First, capture the design intent of an artefact (including functional description, geometric or assembly constraints, performance criteria). Second, record the design process (the communications among the individuals and parties involved in design process, the decision-making that occurs, as well as the decision-making process).

2.5 Design Rationale Capture and Editing Tools

Capturing design rationale has been a research topic for several decades. Inspired by simple, but yet powerful argumentation based concept IBIS, research community triggered the development wave of derivative DR capture systems. Three major DR models can be identified (Ishino and Jin, 2002): argumentation-based design rationale, action-based design rationale, and model-based design rationale. The most noted DR capture methods are presented in the following sections.

2.5.1 Argumentation-based Design Rationale

2.5.1.1 Issues-Based Information System

The first attempt to use rationale as a decision supporting tool can be traced back to 1970. Kunz and Rittel (1970) proposed IBIS approach which intended to address ill-structured problems and support political and social decision processes. The method was recognized in the software engineering industry and was further developed as an aid to engineering design processes (Lee and Lai, 1991; Conklin and Begeman, 1987; Yakemovic and Conklin, 1990). IBIS is built on the concept of argumentative process and is represented as a graphical network of problems with alternative solutions and arguments. An Issue is raised in the form of question, problem or concern which triggers the discussion in an engineering design process. A Position is generated to resolve the Issue with the Arguments linked to it, which either support that Position or object to it.

gIBIS is an enhanced version of IBIS. Conklin and Yakemovic (1991) defines gIBIS as a graphical hypertext software tool for building IBIS networks. The method introduces the ‘generalize/specialize’ relations that can be linked to Arguments as well as to Positions. In addition, external objects, such as documents or sketches, can be represented and linked to relevant nodes (Lee and Lai, 1992). Nodes and links are annotated with the informal rhetorical information (Lubars, 1991). An example representation of gIBIS method is shown in Figure (2.7).

2.5.1.2 Procedural Hierarchy of Issues

McCall (1991) developed the Procedural Hierarchy of Issues (PHI) method which further extended IBIS approach. PHI method is based on Issue-Serve Systems theory, a descriptive theory of design (McCall, 1986; McCall, 1991), which uses an additional argumentative process and broadens the definition of Issue. *‘Issue-Serve Systems theory states that the design process is a quasi-hierarchical system of question answering processes in which the question-answering processes are related to each other by inter-issue dependencies called serve relationship.’* (McCall, 1991: p. 31). Adopting the serve relationship eliminates the variety of relationships used in IBIS. Therefore, the complexity of the inter-issue structure with relationships are reduced leading to a simple tree-like network of issues connected by serve relationships only (McCall, 1991) (Figure 2.8). Moreover, the PHI method redefined some of the IBIS data structures as well as introducing the decomposition process (McCall, 1991). For example, the Position is defined as an Answer, where construction of a structure of sub-Answers further specifies the solution for the raised Issue. Similarly, sub-Argument can be structured under Argument, which further augments its parent Argument. The process of decomposition of

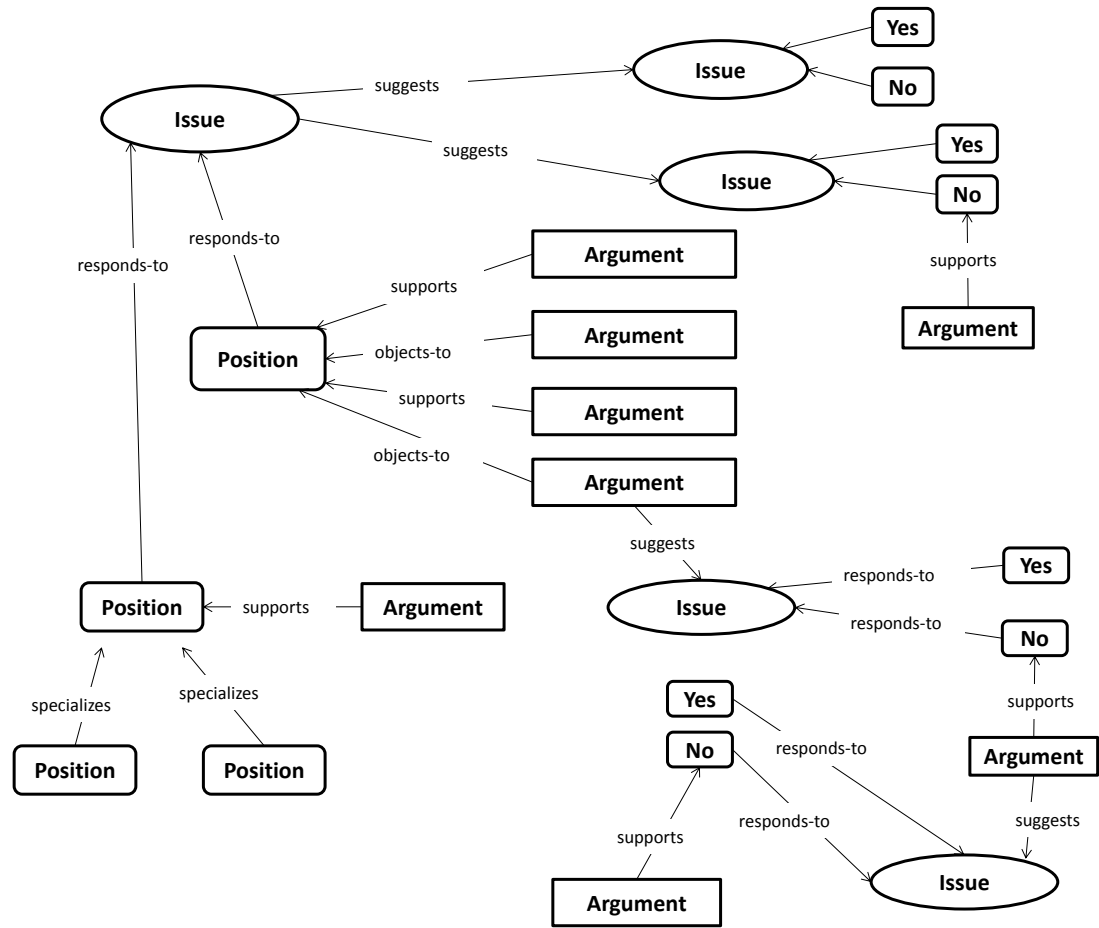


Figure 2.7: gIBIS example. Adapted from [Lee and Lai \(1992\)](#).

the parent Issues, Answers or Arguments into hierarchical multi-level structures of sub-Issues, sub-Answers and sub-Arguments, provides the designer with more options for the resolution of the raised Issue (Figure 2.8). In addition, the specificity and granularity of sub-relations is expressed.

2.5.1.3 Decision Representation Language

Decision Representation Language (DRL) is a language developed by [Lee and Lai \(1991\)](#) to represent and manage the qualitative elements of decision making (i.e. decision rationale). It encompasses five design spaces to support design tasks; argument space, alternative space, evaluation space, criteria space and issue space ([Lee and Lai, 1992](#)). In the argument space, arguments are represented as a set of related Claims that can be expressed as rules, assumptions, statements or facts. All DRL relations are special types of Claims that can be argued about by supporting, denying or qualifying them ([Lee and Lai, 1991](#)). The criteria space is represented by Goals. The alternative space, as the name suggests, includes the alternative solutions. In the evaluation space attributes are

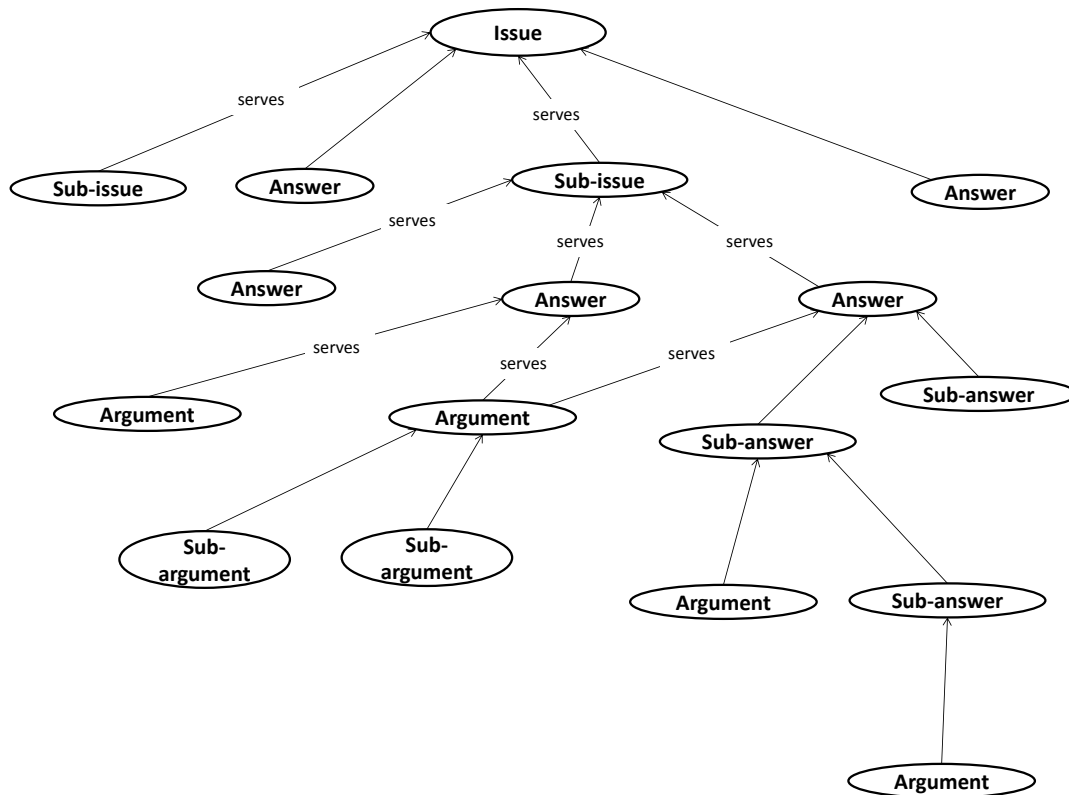


Figure 2.8: PHI model. Adapted from [Chan \(2007\)](#).

assigned to Claims which evaluate the importance of the Claim and identify the plausibility and degree of the Claim. In DRL, the issue space is represented by a Decision Problem. An example of the DRL model is illustrated in Figure 2.9. [Lee and Lai \(1991\)](#) argues that characterising the domain of design rationale is important, as it identifies the rationale elements and the relations between them. Therefore, DRL can be seen as a method that defines the scope and evaluates the expressive adequacy of design rationale representations.

2.5.1.4 Questions, Options, and Criteria

[MacLean et al. \(1989\)](#) uses a concept of design space to represent the design rationale. *'Design Space Analysis uses a semiformal notation, called Questions, Options, and Criteria (QOC), to represent the design space around an artefact'* ([MacLean et al., 1991](#): p. 201). Furthermore, it creates the representation of a structured space of design alternatives with corresponding criteria that justifies the selection of a particular artefact. In QOC, Questions refer to key issues in the design space, Options are alternative answers to Questions and Criteria are the pros and cons of the possible Options. Moreover, to emphasise the positive and negative relationships between Options and Criteria in the design space the two constructs are linked by solid and dotted lines, respectively (Figure 2.10). According to [MacLean et al. \(1993\)](#), the purpose of QOC is not to capture the

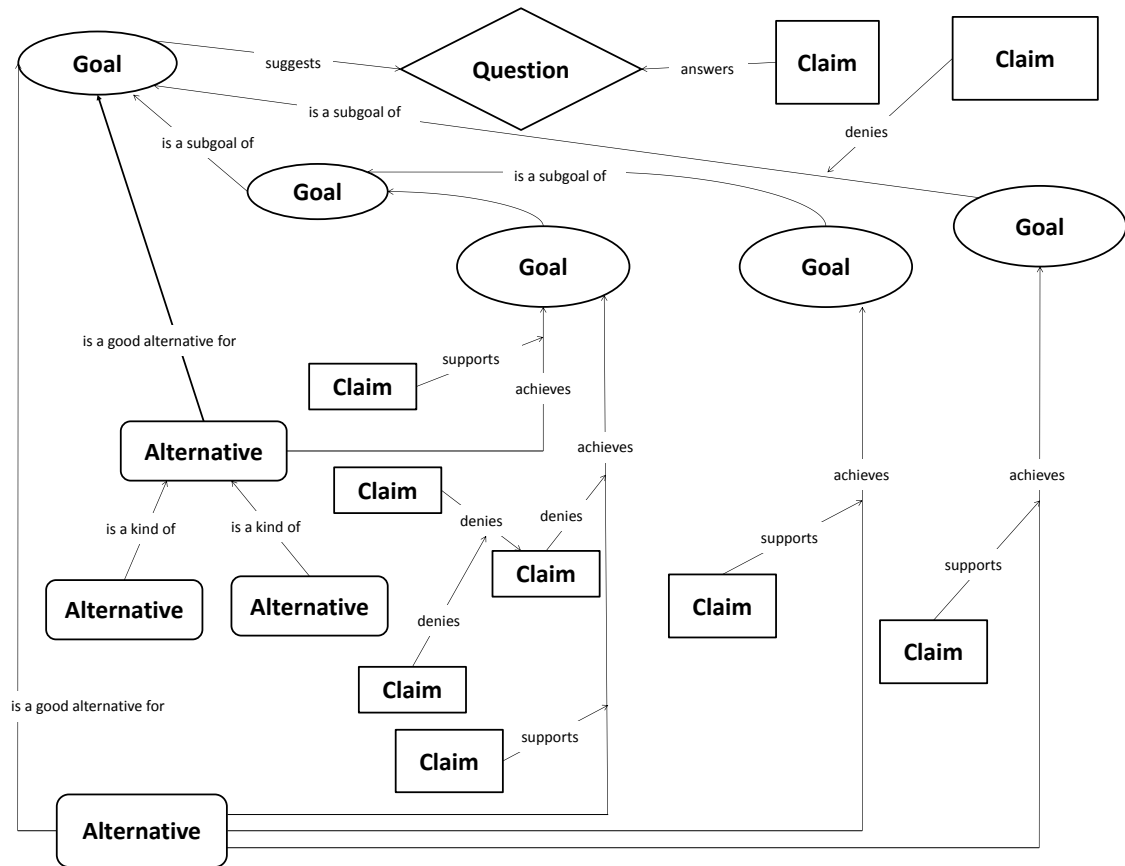


Figure 2.9: DRL model. Adapted from [Lee and Lai \(1991\)](#).

history of design deliberation, but rather to represent explicitly the design space around an artefact with the alternative design options structured by questions and the reasons for choosing among those options.

2.5.2 Derivative Design Rationale Systems

The increase in rationale research activities occurred when hypertext systems became more popular ([Burge, 2008](#)). As a result, new prototype systems were developed for design rationale capture. Table 2.1 lists some prototype systems that are based on the methods described in the previous sections.

2.5.3 Graphical Design Rationale Editors

Despite the huge variety of design rationale capture methods developed in research environment only a few have been successful. Two principal graphical design rationale editors can be identified: Compendium ([Buckingham Shum et al., 2006](#)) and Design

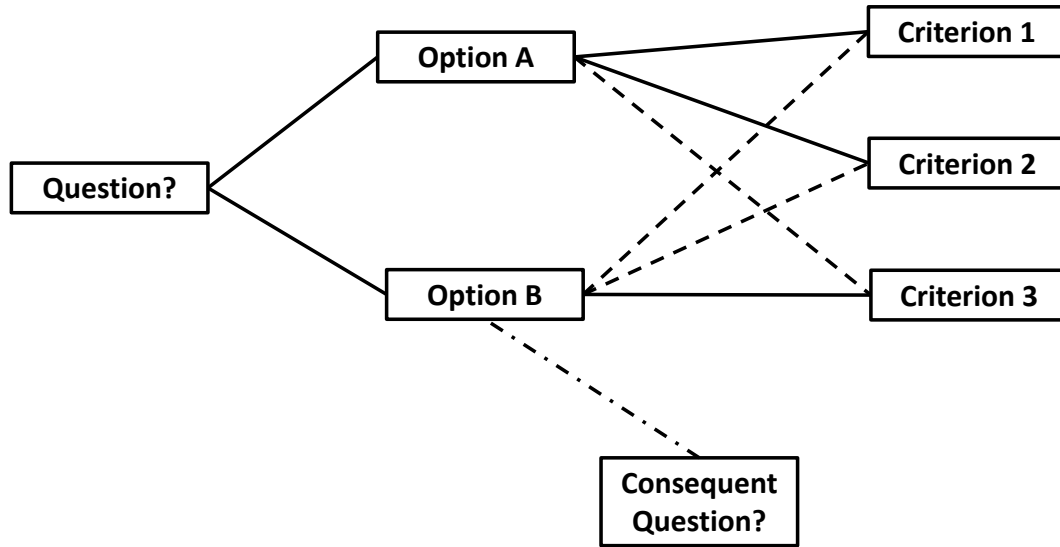


Figure 2.10: The components of a design space represented by QOC notation.
Adapted from [MacLean et al. \(1993\)](#).

System Name	Knowledge Representation	Design Domain	Year
VIEWPOINTS (Fischer et al., 1989)	IBIS	Kitchen	1989
JANUS (Fischer et al., 1989)	PHI	Kitchen	1989
SIBYL (Lee, 1990)	DRL	Generic	1990
ADD (Garcia and Bicharra, 1992)	Argumentation & Model-based	HVAC (Heating, Ventilation and Air Conditioning)	1992
REMAP (Ramesh and Dhar, 1992)	IBIS	Generic	1992
HOS (Shipmann and McCall, 1997)	PHI	Generic	1997
PHIDIAS (Shipmann and McCall, 1997)	PHI	2D, 3D	1997
DRARS (de la Garza and Ramakrishnan, 1995)	QOC	Building	1995
C-DeSS (Klein, 1997)	IBIS, PHI, DRL	Geometry	1997

Table 2.1: Prototype design rationale systems.

Rationale Editor (DRed) ([Bracewell et al., 2009](#)). Compendium has been used on a number of different projects and is being developed further as an open source design rationale editor. DRed has been positively evaluated by the designers at Rolls-Royce and has been accepted and integrated into the company’s Product Life-cycle Management toolset. The brief description of the tools is given in the following two sections.

2.5.3.1 Compendium

Compendium was originally developed to aid cross-functional business process redesign teams in managing design ideas, issues, tracking design rationale and managing the project’s overall structure and goals ([Selvin and Buckingham Shum, 2002](#)). Compendium evolved into an open source graphical design rationale editor which is based on the gIBIS and QOC design rationale representation concepts ([Buckingham Shum et al., 2006](#)). [Buckingham Shum et al. \(2006: p. 114\)](#) recognised the ‘intrusiveness’ problem of the tool, however, authors also acknowledged that “*deeper understanding of a domain comes through the discipline of expressing knowledge within a structural framework, working*

to articulate important distinctions and relationships. [...] Design rationale that yields insight into the complex ideas and arguments that may lie behind a decision does not come ‘for free’: effort must be invested at some point in the rationale management lifecycle.” Compendium is mainly aimed at bridging the gap between synchronous, face-to-face interaction and asynchronous, distributed interaction (Selvin et al., 2001). In other words its purpose is to capture the knowledge and decisions made in the meetings and then to structure and distribute that knowledge between the respective professionals. This knowledge will later serve as a reference when tackling ill-structured problems. Compendium, as with many other argumentation based methodologies, uses the concept of issues, solution and arguments. The main elements (nodes) are shown in Figure 2.11. According to Selvin et al. (2001), Compendium enables groups to collectively elicit, organize and validate information which is necessary for tackling ill-structured problems.

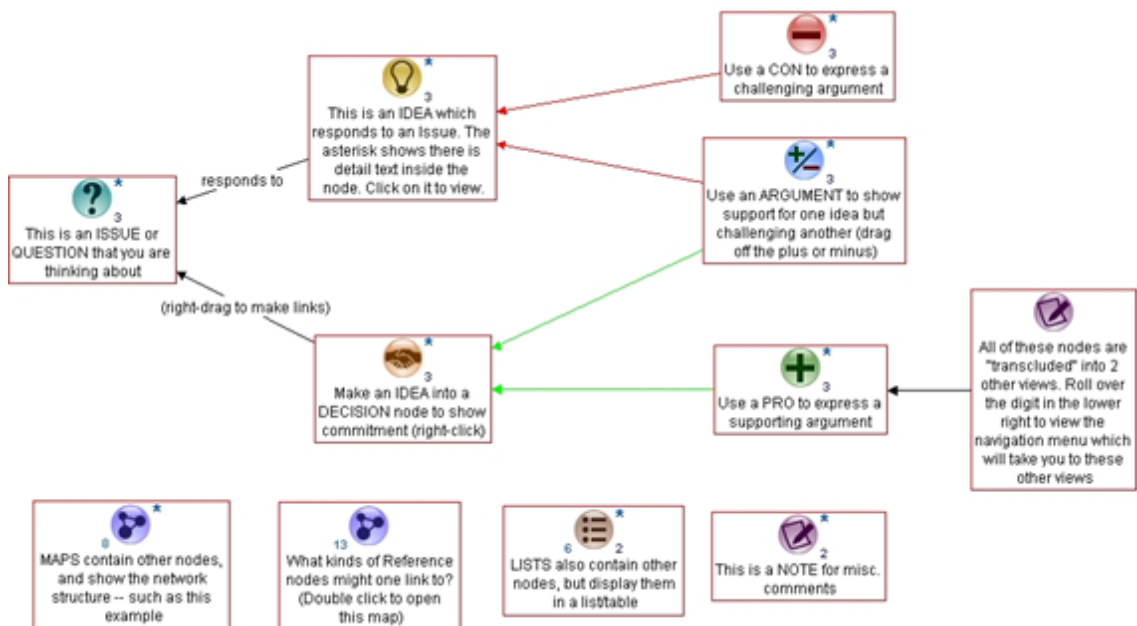


Figure 2.11: Compendium node types and argumentation structure, an example from the Compendium user instructions.

2.5.3.2 Design Rationale Editor

Design Rationale Editor (DRed) originated from existing open source application Graphlet (Himsolt, 2000), which was progressively customised evolving into a tool to support design rationale capture (Bracewell et al., 2009). The DRed is an IBIS derivative recently further optimised to map an integrated information space covering product planning, specification, design, and service (Auricchio and Bracewell, 2013). The tool has been developed in collaboration with Rolls-Royce plc, and is owned and controlled by the company (Auricchio et al., 2013).

As claimed by the authors [Bracewell et al. \(2009: p. 173\)](#) DRed “*is a simple and unobtrusive software tool that allows engineering designers to record their rationale as the design proceeds*”. Furthermore, the issues, options and associated pro and con arguments are captured and presented in the form of a directed graph of dependencies. The DRed elements from a predefined set of types are created, positioned and linked manually by the user. Main elements and statuses of the DRed are illustrated in Figure 2.12 and consists of Issue, Answer, Pro, Con, Text, Task and File ([Auricchio and Bracewell, 2013](#)).

The following additions to the DRed software tool made it more usable and accepted by designers comparing to other design rationale capture tools:

- Integration of colour coding and statuses.
- Introduction of tunnelling links, which allow bidirectional hyperlinking and eliminate the need of dedicated Database Management System.
- Introduction of tunnelling links with external documents.
- Introduction of nodes that display an arbitrary number of lines of text.
- Allowing the text to overlay coloured graphics that signify the node types and statuses.

DRed has gained high acceptance within Rolls-Royce and is now part of their standard PLM toolset ([Bracewell et al., 2009](#); [Auricchio and Bracewell, 2013](#)). It has to be noted, however, that DRed is mainly used in concept and preliminary design stages encompassing four main application areas, namely diagnosing a problem (Figure 2.13), designing a solution, completing a standard checklist template and communicating the final design and its rationale ([Bracewell et al., 2009](#)).

2.5.4 Action-based and Model-based Design Rationale

2.5.4.1 Design History Tool

The Design History Tool (DHT) ([Chen, 1991](#); [Nagy et al., 1992](#)) stores structured and hierarchical representations of a design based on designers actions during the design process as well as on verbal and written data captured during collaborative design processes. The six data elements: *design-object*, *constraint*, *issues*, *proposals*, *arguments*, and *decisions* are used to represent design information in the DHT computer data base (Figure 2.14).

The design process is characterised by these six data-elements which are linked to

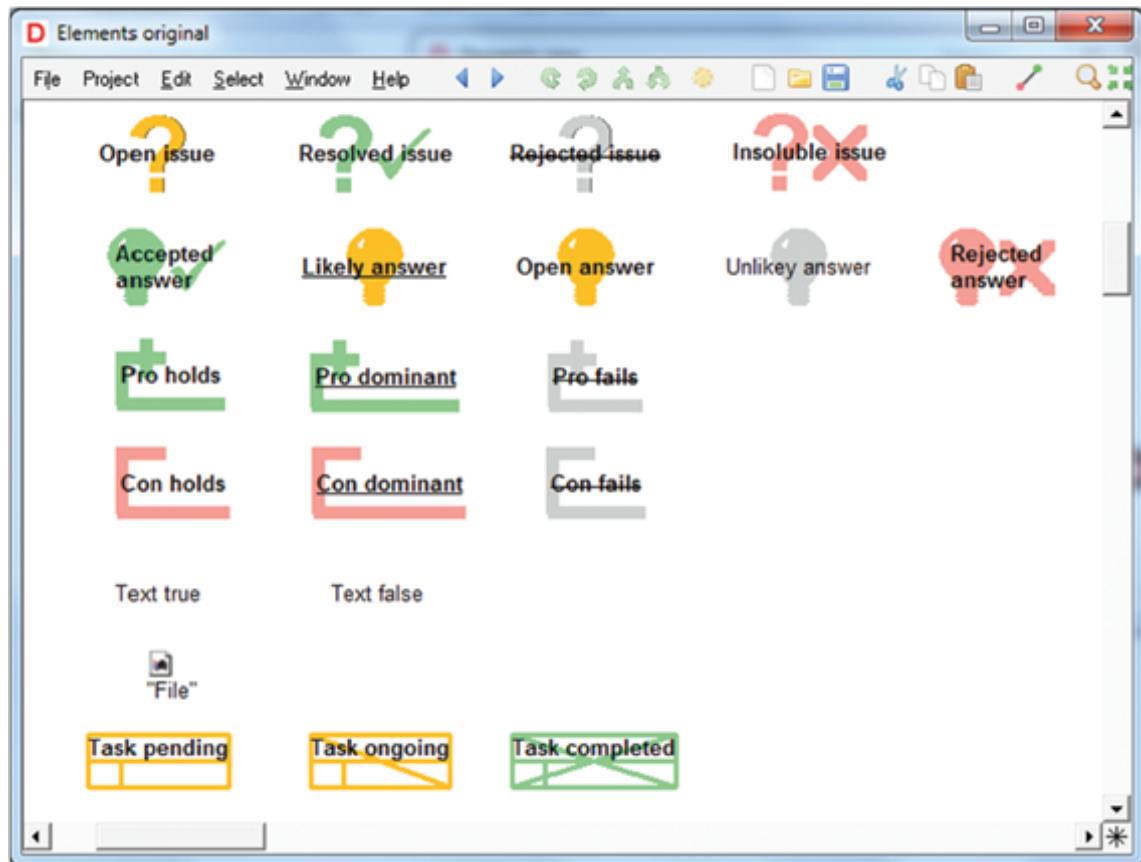


Figure 2.12: DRed elements and statuses (Aurisicchio and Bracewell, 2013).

form higher level data-structures: *decision-chain*, *decision-process*, *issue-decomposition*, *issue-network* (Nagy et al., 1992). The *decision-chain* network is used to capture the chronological history of the design process (Figure 2.15); the *decision-process network* is used to record the decision history to solve issues as they arise (Figure 2.16); and the *issue-decomposition* and *issue-network* are used to represent the interconnectivity of the design issues. The issues can be decomposed into sub-issues that are interlinked with the ‘and links’ (refer to Figure 2.17). The parent issue is declared as solved only when all child issues are solved.

2.5.4.2 Grammar and Extended Dynamic Programming

In order to capture the design process knowledge (referred to as *know-how* knowledge), Ishino and Jin (2002) proposed a three-layer design process model to represent generic design processes, and a Grammar and Extended Dynamic Programming (GEDP) method that captures the actions designers took, using a CAD system, during the design process. The recorded design-history is then identified and translated into know-how knowledge. This method was developed to eliminate the disruption to the designer’s work during the design process. In this proposed method knowledge is extracted from the events

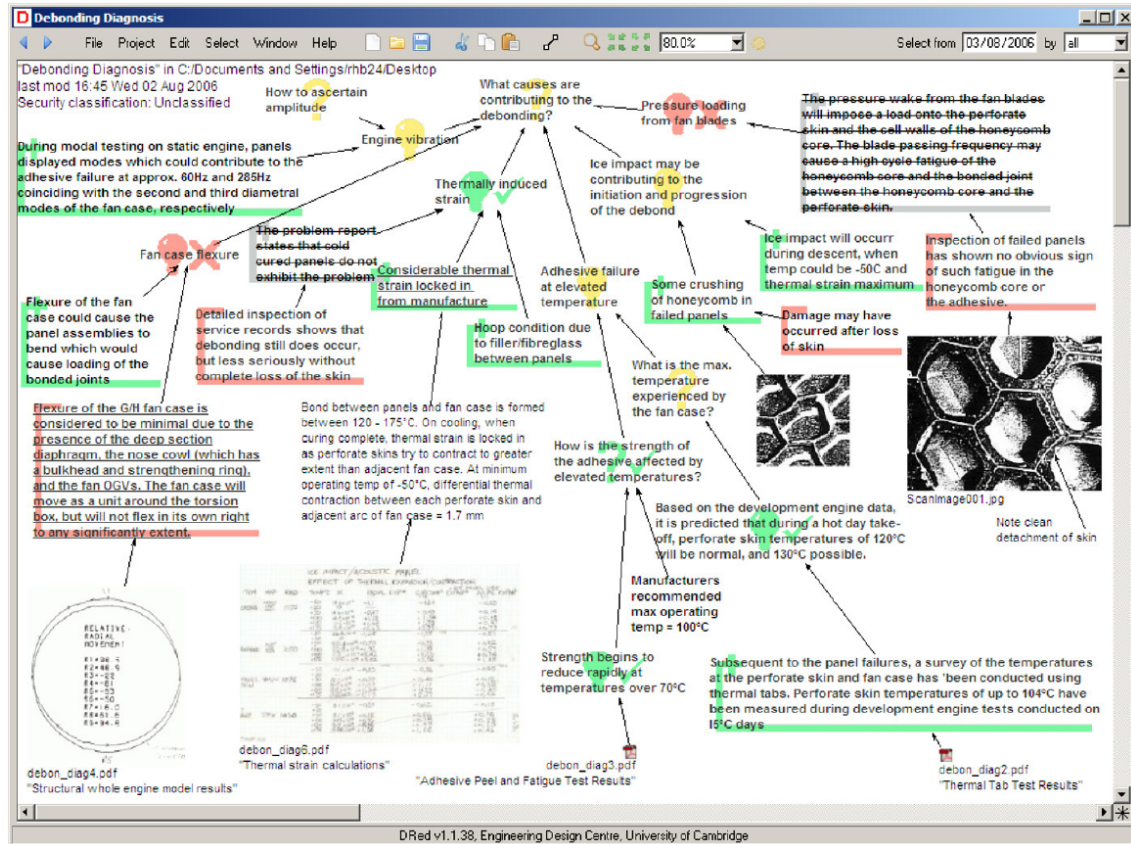


Figure 2.13: DRed example. Diagnosing a problem (Bracewell et al., 2009).

and sequences of operations designers performed during the design process. The design process is structured into three layers: Event-Layer, Operation-Layer, and Product Model-Layer. The events and operations are grouped and structured by the GEDP algorithms using the predefined rules. Design operations in the Operation-Layer are generated from the multiple design events found at the Event-Layer. The alternative designs reside in the Product Model-Layer and are generated from the multiple design operations (Figure 2.18). In order to represent the development process of design, a hierarchical structure of product models are created (Figure 2.19). According to Ishino and Jin (2002), the hierarchical tree structure can be used by designers as a backtracking mechanism to locate alternative designs which were not developed further. These alternative designs could be the starting point for future product developments. Figure 2.19 shows the sequence and the evolution of the product during the design process where P₉ was selected as a final product model that satisfies all the requirements.

The advantage of the proposed model and methods is that it captures the *know-how* knowledge without disrupting the designer's normal design process. However, a new problem of information management is created due to large volumes of data being recorded which has to be managed in an effective and efficient way. Furthermore, the method is oriented explicitly towards *know-how* knowledge, which is about *design procedures*, but not towards *know-why* knowledge, which refers to *rationale*.

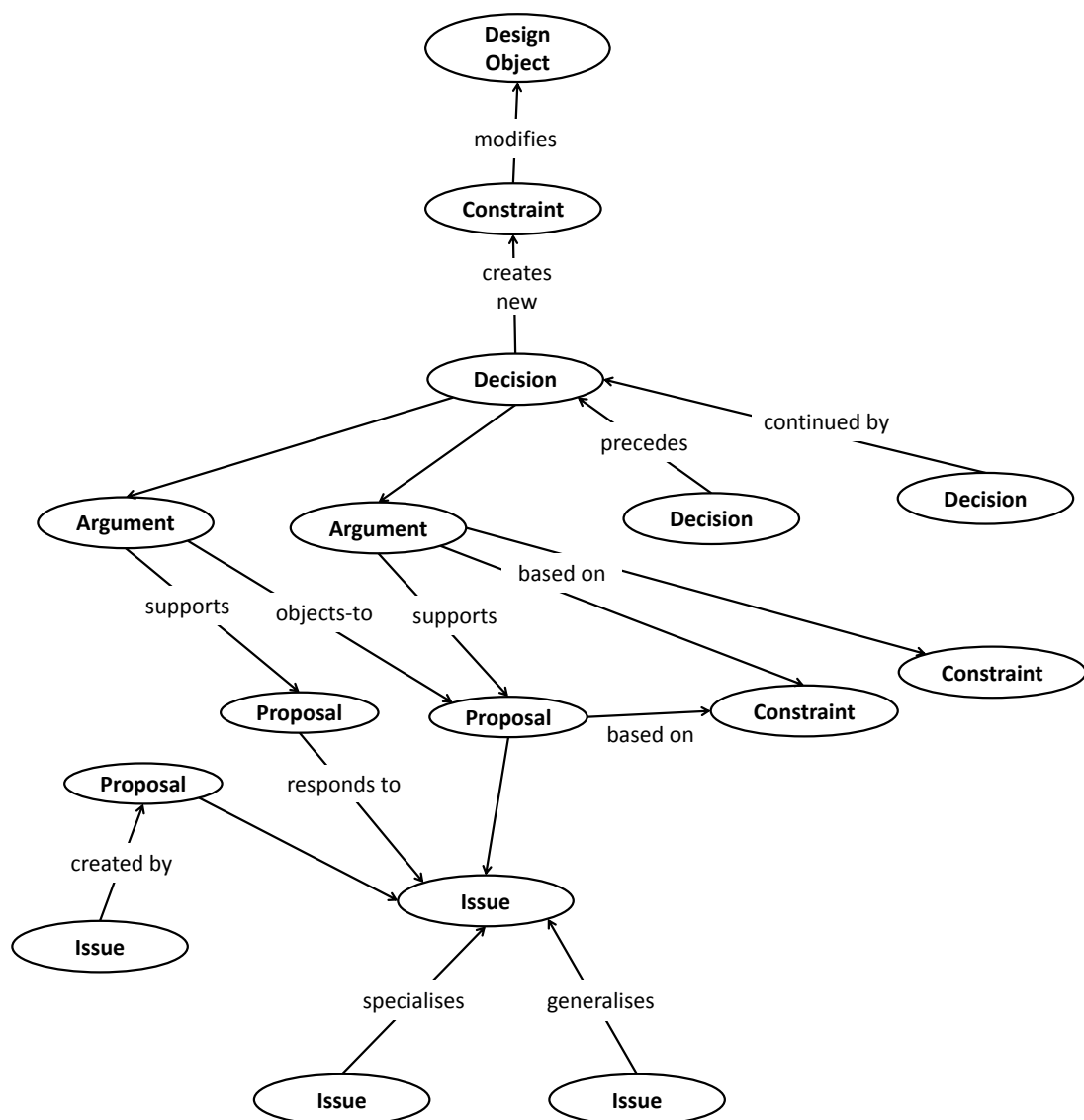


Figure 2.14: The DHT model. Adapted from [Wiegeraad \(1999\)](#).

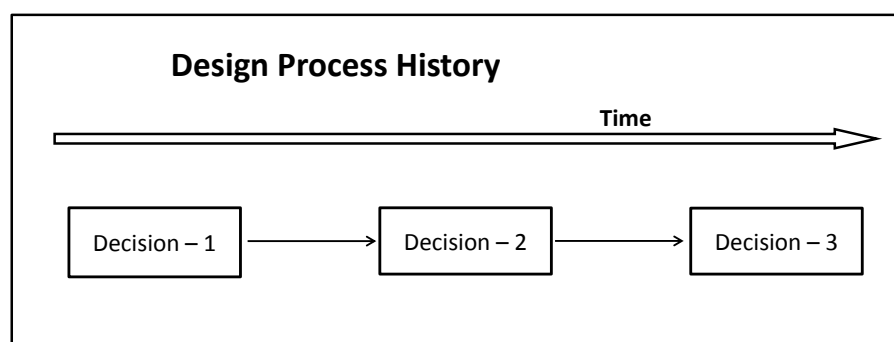


Figure 2.15: Decision-chain ([Nagy et al., 1992](#)).

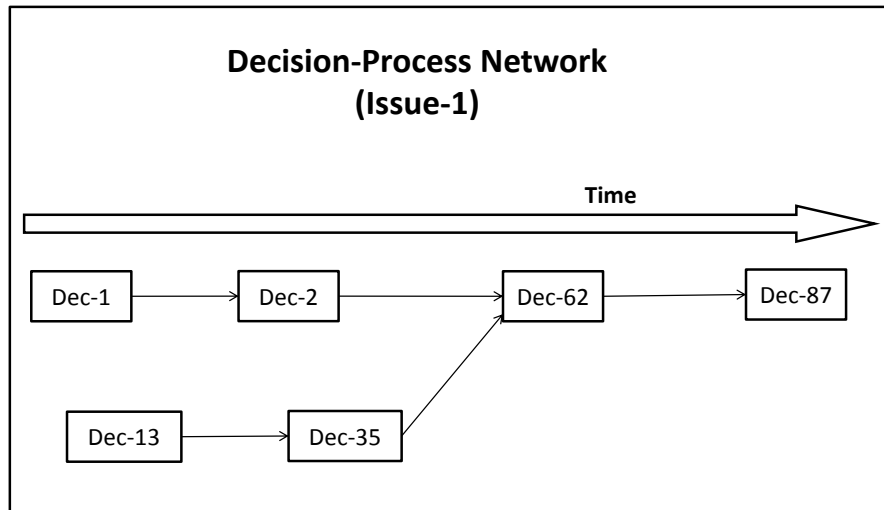


Figure 2.16: Decision-process (Nagy et al., 1992).

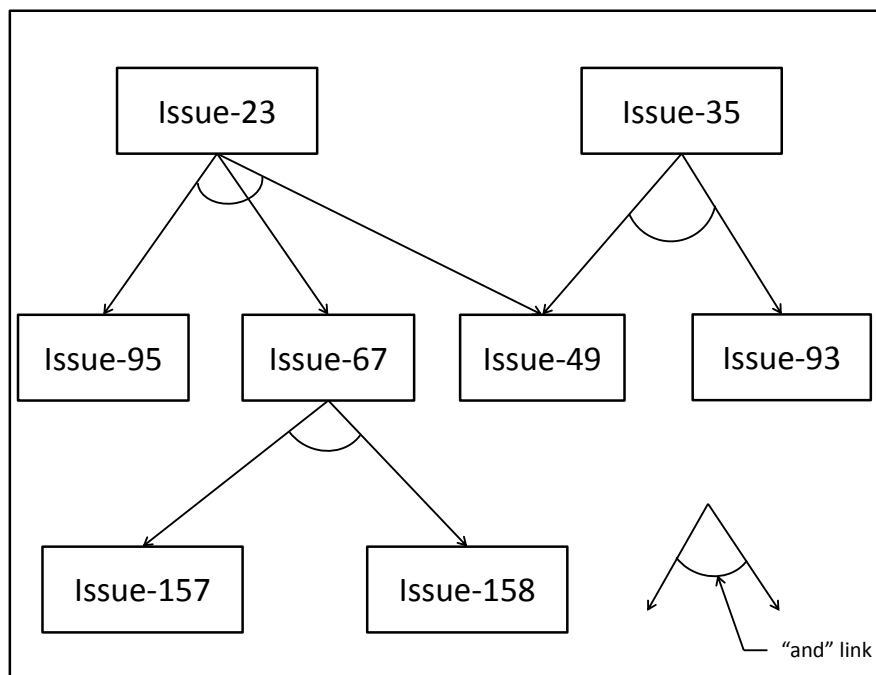


Figure 2.17: Issue-decomposition (Nagy et al., 1992).

2.5.4.3 Rationale Construction Framework

Rationale Construction Framework (RCF) developed by Myers et al. (2000), is an experimental system for rationale information acquisition that captures design process rationale without disrupting a designer's normal activities. Similarly to GEDP (Ishino and Jin, 2002), the design process history from designer's interactions with a CAD tool

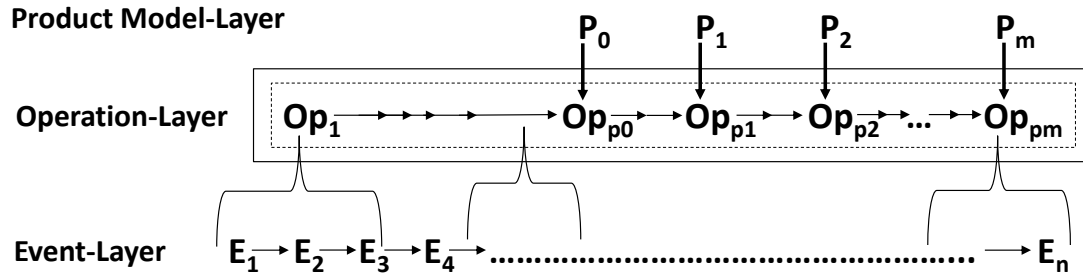


Figure 2.18: A three layer design process model (Ishino and Jin, 2002).

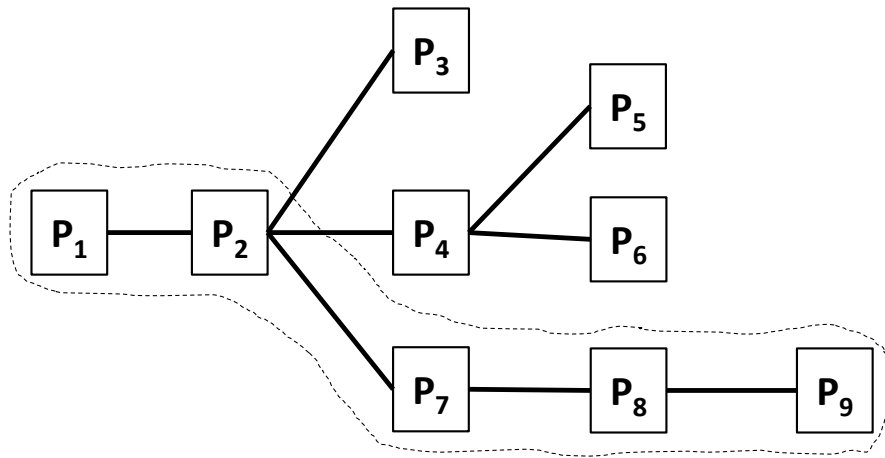


Figure 2.19: The hierarchical tree structure of product models: P_1 - initial product model; P_9 - final product model that meets the requirements (Ishino and Jin, 2002).

is recorded and the data is structured. The underlying difference between the two methods is that in RCF predefined rules, called *design metaphors*, and qualitative reasoning are used to capture the design process, whereas GEDP uses complex pattern recognition to detect design procedures and by applying rules derived from design principle knowledge procedure features are obtained. The main focus of this approach is on *know-how* knowledge, leaving *know-why* knowledge out of the equation.

2.6 Design Rationale Capture During Detailed Design Stage

According to the literature, the majority of research is focused on Design Rationale capture in *concept* and *preliminary design* stages. Only a few studies, however, looked at Design Rationale capture in *detailed design* stage. In fact, there are only a few published empirical studies with quantitative results on design rationale capture during the design process (Conklin and Begeman, 1988; Conklin and Yakemovic, 1991; Van Schaik, 2013).

The authors that conducted the empirical studies reported the low design rationale capture during embodiment and detailed design stages. The recent empirical study on design rationale capture during the UAV design process, conducted by [Van Schaik \(2013\)](#), revealed that very little rationale was captured during the detailed design phase. Conversely, a substantial amount of design rationale was captured during the concept design stage. The study confirmed the results reported in the literature as well as demonstrating that no improvements over the last two decades have been made to aid the design rationale capture in the detailed design stage. [Van Schaik \(2013\)](#) identified three factors, namely motivational, environmental and technical, that are responsible for low design rationale capture during geometry design:

- Lack of structured methods makes it difficult to link rationale to part geometry - technical factor.
- Often design activities in the detailed design phase are performed under time pressure, leaving little time (or no time) for design rationale capture - environmental factor.
- Tools that are not integrated into the design process are deemed intrusive, disrupting the normal design process - environmental factor.
- Designers do not see the benefit of capturing rationale during geometry design phase or they think the effort is not justified - motivational factor.

The conclusion can be drawn that design rationale capture in the detailed design stage is not well supported by the current tools. Although environmental and motivational factors contribute to the low design rationale capture during geometry design, it seems that the most limiting factor is technical.

2.7 Use of Annotations in Mechanical Design Context

Annotation in [Oxford \(2010\)](#) dictionary is defined as “*a note by way of explanation or comment added to a text or diagram*”. In general, annotation can be interpreted as extra information attached to a particular point in a document or other information object ([Li et al., 2009](#)). Historically, annotations have been used as in-context information storage. The annotation techniques were adopted by the design research community to aid collaboration and communication during the design process. Due to the fact that geometry models are unable to contain contextual information such as design decisions and rationale, the digital annotation of geometry models is an active area of research investigating the benefits of annotations in a collaborative design environment ([Lenne et al., 2009](#)). [Li et al. \(2009\)](#) argues that digital annotations improve the efficiency of collaboration between the parties participating in the product development process as well as contributing to information management by providing the storage medium,

retrieving and interpreting information. The characteristic elements of annotations in a mechanical design context can be summarised as follows (Guibert et al., 2009):

- the document is the target that the annotation refers to,
- the content of an annotation is the information the annotation imparts,
- the anchor of an annotation is a particular point in the document or other object where the annotation is attached,
- the annotations are private or public,
- the life-span of an annotation may be short-term (i.e. “post-it” sticker) or permanent (i.e. the footnotes of a document),
- the annotator and the user of an annotation may be different.

The document or other object that the annotation is attached to provides the necessary context to the annotation content which includes, but is not limited to, raw text, Extensible Markup Language (XML), Resource Description Framework (RDF), the Multimedia Content Description Interface from Moving Picture Experts Group (MPEG 7) (Li et al., 2009). Several authors used annotation techniques:

- to represent graphical knowledge (Aubry et al., 2007),
- to record design review outcomes (Hisarciklilar and Boujut, 2007),
- to reduce communication ambiguity (Hisarciklilar and Boujut, 2009),
- to speed up geometry editing (Alducin-Quintero et al., 2011),
- used annotations in collaborative design (Lenne et al., 2009),
- to clarify design intent (Li et al., 2011),
- used as a medium to contain design decision information (Boujut and Dugdale, 2006), and
- contain or link to design rationale (McKay et al., 2009).

However, Hisarciklilar and Boujut (2009) argues that “*serious effort is still needed in order to understand the actual role of annotations to support argumentative design communication*”. Li et al. (2009) reports that there are some issues in content organisation as well as retrieval and interpretation of annotations. The author also emphasises the fact that there is no well-recognised method to represent anchors to attach structured content and that more efficient support for external and internal data exchange is necessary.

Although there are a few issues that need to be resolved, the large number of published papers proved that digital annotations can assist with communication in the design process and especially with a collaborative design environment.

Summary

This section has explored the definitions of Design Rationale and presented the comprehensive list of the most noted Design Rationale capture and editing methods and tools. Three major DR classes have been identified and presented: argumentation-based design rationale, action-based design rationale, and model-based design rationale.

The main DR goals in the product development process are ([Szykman et al., 2001](#)): to capture the design intent of an artefact (including functional description, geometric or assembly constraints, performance criteria); and to record the design process (including the communications among the individuals and parties involved in design process, the decision-making that occurs, as well as the decision-making process).

According to the literature, only a few studies looked at Design Rationale capture in the detailed design stage. The recent empirical study on design rationale capture during the UAV design process, conducted by [Van Schaik \(2013\)](#), revealed that very little rationale was captured during the detailed design phase. The study concluded that lack of structured methods how to link rationale to part geometry appears to be the main limiting factor for the low design rationale capture during geometry design.

Geometry Representation and Modelling

2.8 Mechanical Engineering Design Support Systems

Generally, activities in mechanical engineering design are supported by knowledge-based systems, CAD/CAM systems, PDM and PLM systems ([Regli et al., 2000](#)). These systems have been developed to aid the creation and management of design information. For example, the feature-based parametric CAD tools are able to record the design sequence of the artefact where the design intent is captured as a result. The PDM and PLM systems significantly improved the management of design information by providing a platform for efficient structuring, identification and classification of the products, assemblies and components. Knowledge-based engineering, artificial intelligence, and expert systems have been developed to record the history of the design process. The knowledge captured during the design process is formalized in the form of rules and maintained in a separate knowledge-base. The information recorded by these systems is constantly updated and reused in future projects.

Mechanical design support systems have changed engineering design practice and improved the geometrical modelling aspect of the design process ([Ullman, 2002](#)). However

the knowledge capture side is still an open question which requires further research in order to make the design process more efficient. The current CAD tools capture the geometric relations and geometric design intent but fail to model the actual decision structure or rationale for captured relations. The PDM and PLM systems improved information management and reuse. However the main focus of these systems is towards reusing product geometry and documentation. In other words, these systems tend to contain the information that describes *what* the product is but not the information of *why* the product was designed in a particular way (Chan, 2007). The knowledge-based systems (KBS) and expert systems require continuous maintenance due to constant evolution of knowledge. This requires significant time resources of the designer to update and manage these systems. Moreover, these systems are often developed as separate systems and are not integrated into CAD tools or PDM and PLM systems. As a result, the design time is compromised by the time required to access the knowledge systems and search for required information.

2.9 The Specific and Generic Systems

Mechanical design support systems discussed in the previous section can be classified as specific or generic systems (Figure 2.20).

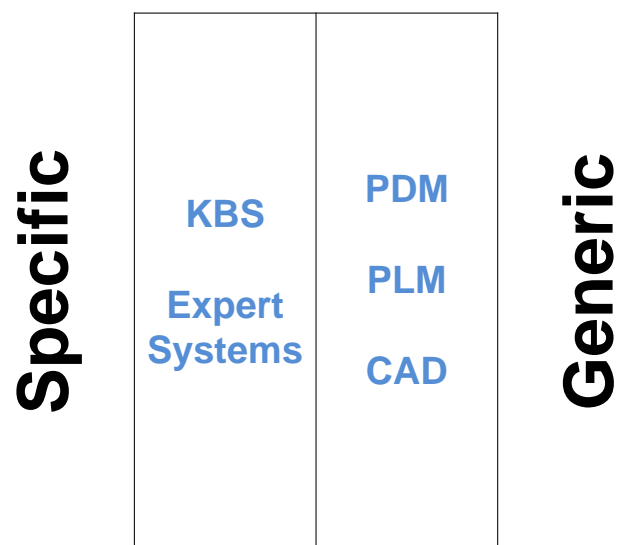


Figure 2.20: Specific and Generic systems.

For example, knowledge-based systems and expert systems fall into the category of specific systems as they are often domain and product specific systems developed for particular processes and products and are not applicable to other processes or products. On the other hand, CAD systems and PDM and PLM systems can be classified as generic systems. These systems have been developed to aid different areas of the design process (geometry modelling, design information management, etc.) but not the rationale capture.

2.10 Geometry Representation

Computer based geometric modelling started with 2D wireframe geometry representation scheme and later progressed to 3D wireframe, to surface modelling, to solid modelling and to feature-based geometry representation schemes. In the wireframe models geometry is characterised as lines and points connected in a certain way to represent a geometric shape. The surface modelling is another type of geometry modelling scheme that uses surfaces and curves to represent the geometry. It is based on Non-Uniform Rational B-Splines, or otherwise known as NURBSs, and Beizer mathematical techniques (Davies, 2008). The solid modelling scheme is a synthesis of wireframe and surface modelling techniques. The solid is defined as nodes, edges, and surfaces connected in such a way that an enclosed and filled volume is constructed. Constructive Solid Geometry (CSG) and Boundary Representation (B-rep) are two main methods used to represent solid geometry. The underlying principle of CGS is based on the primitive 3D solid geometric shapes that are combined or subtracted from each other using Boolean operators in order to construct a more complex geometry (Davies, 2008). In the B-rep representation method, the three-dimensional geometries are represented in terms of their spatial boundaries. In other words, a solid is represented as a collection of vertices, edges, and surfaces of a volume, where each face of the surface is bounded by a ring of edges meeting at vertices (Davies, 2008). Modern CAD systems utilise a hybrid approach to represent the geometry. Feature-based parametric modelling techniques leverage the methods described above. This modelling technique enables designers to produce more flexible designs based on design variables and parametric features. The part is described as a sequence of features in CAD's procedural representation tree, which are controlled by design variables. According to Shih (2014: p. 1-6) "*the concept of parametric features makes modelling more closely match the actual design-manufacturing process than the mathematics of a solid modelling program*".

The next section defines a term feature and describes three types of features discussed in this work.

2.11 Geometry Modelling

Geometry modelling is a core activity of the design process during which the geometric features are created. Commercial feature-based CAD systems, such as Siemens NX, Catia V5 and Solidworks are used for geometry modelling where low level geometric information is converted into a high level description of the product in terms of form, functional, manufacturing or assembly features (Di Stefano et al., 2004). In general, features can be described as a medium of information transmission that carries rich semantic and geometrical information about a set of geometrical entities associated with a specific part or a region of the model (Gao et al., 2004; Zhang and Luo, 2009).

However, feature definitions depend on different application viewpoints. Three types of features are discussed in this work, and are defined below:

Modelling Feature is an intermediate operation in the procedural order of the modelling tree which is necessary to create the design, or manufacturing, or assembly feature.

Design Feature is a set of modelling features that represent an abstract geometry related to a component's function, design intent, or model construction methodology (Gao et al., 2004).

Manufacturing Feature is a feature created by a specific manufacturing process or operation (i.e. turning features, milling features, and drilling features). It can be a single design feature, if it can uniquely represent it, or a combination of several design features (Anjum et al., 2012).

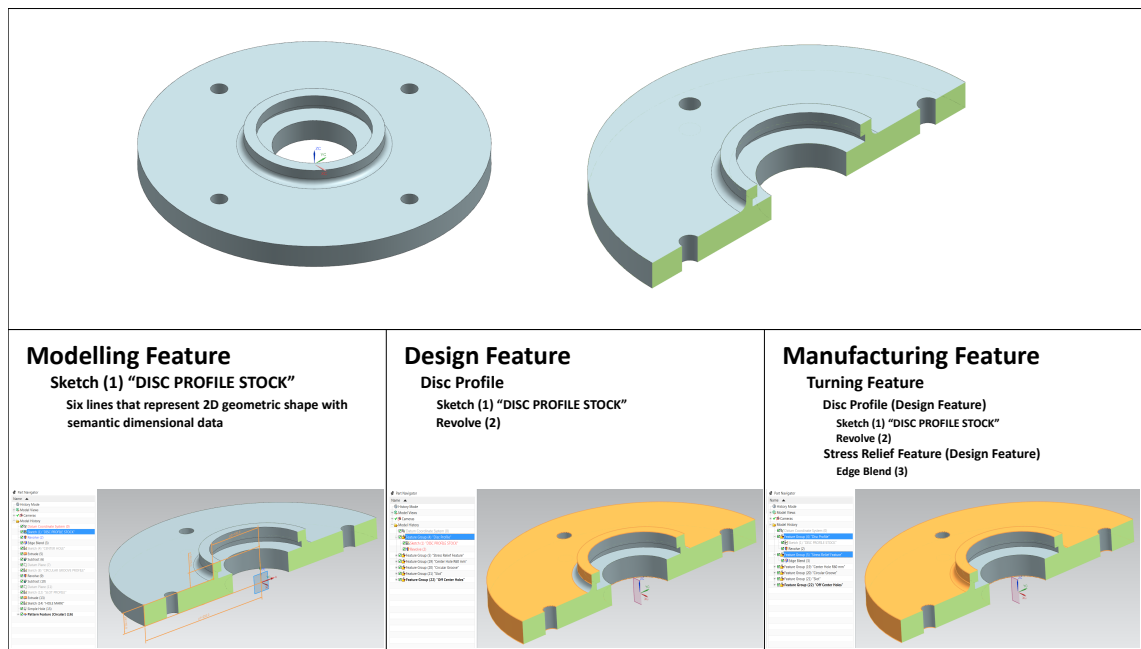


Figure 2.21: Feature Types.

Figure 2.21 illustrates three types of features. As seen in the bottom left image in Figure 2.21, six lines that represent 2D geometric shape (i.e low level geometric information) define one of the modelling features (*'Sketch (1) DISC PROFILE STOCK'*); in the middle image two modelling features (*'Sketch (1) DISC PROFILE STOCK'*, *'Revolve (2)'*) define one of the design features (*'Disc Profile'*); in the bottom right image two design features (*'Disc Profile'*, *'Stress Relief Feature'*) define the turning feature. The fact that the feature definition depends on different application viewpoints, hampers the immediate usability of the feature-based model across different domains in the product development lifecycle. For example, feature sets in the design domain have to be transformed or mapped to the feature sets that are compatible with the manufacturing

domain, before the model can be used in the manufacturing domain. As Figure 2.21 illustrates, the perception of a feature in the design domain is different in the manufacturing domain. Feature recognition is an active area of research (JungHyun et al., 2000; Gao et al., 2004; Jones et al., 2006; Hao, 2014). The aim of a feature recognition algorithm is to recognise feature types from a part geometry using a combination of feature recognition methods: rule-based, graph-based, and geometric reasoning methods (Babic et al., 2008). However, the majority of work done to date developed feature recognition techniques for separate viewpoints, for example, machining feature recognition (Nasr et al., 2014; Zhang et al., 2016), casting feature recognition (Bidkar and McAdams, 2004; Madan et al., 2007), and feature recognition techniques for mould manufacturing (Chen et al., 2011; Jong et al., 2014). It is a complex problem because every manufacturing process has its unique operation sequences and the algorithm that would provide a fully automated feature recognition process for every manufacturing process still does not exist (Babic et al., 2008). Design-by-feature is another approach in attempt to solve the compatibility issue between the design and manufacturing domains. The underlying principle of this approach is based on the instances of feature classes that are defined by a user and stored in a feature library (Bronsvort et al., 2006). The instances of user defined features can be added to the model to accommodate the part manufacturing needs. This is illustrated with an example in Figure 2.21. Two features, namely '*Disc Profile Feature*' and '*Stress Relief Feature*' are defined by the user and combined to create the turning feature.

Summary

Mechanical design support systems have changed engineering design practice and improved the geometrical modelling aspect of the design process. The current CAD tools capture the geometric relations and geometric design intent but fail to model the actual decision structure or rationale for captured relations. These systems tend to contain the information that describes *what* the product is but not the information of *why* the product was designed in a particular way.

Computer based geometric modelling started with 2D wireframe geometry representation scheme and later progressed to 3D wireframe, to surface modelling, to solid modelling and to feature-based geometry representation schemes. Modern CAD systems utilise a hybrid approach to represent the geometry. Feature-based parametric modelling technique enables designers to produce more flexible designs based on design variables and parametric features.

Commercial feature-based CAD systems are used for geometry modelling where low level geometric information is converted into a high level description of the product in terms of form, functional, manufacturing or assembly features. Feature definitions depend on different application viewpoints. The definitions of three types of features (i.e.

modelling features, design features and manufacturing features) have been presented. Feature recognition is an active area of research. The majority of the research developed feature recognition techniques for separate viewpoints: machining feature recognition; casting feature recognition; and feature recognition techniques for mould. It is a complex problem because the algorithm that would provide a fully automated feature recognition process for every manufacturing process still does not exist ([Babic et al., 2008](#)).

Requirements Engineering

Requirements Engineering (RE) is an engineering discipline that deals with the development, elicitation, specification, analysis, and management of the stakeholder requirements ([Wiesner et al., 2015](#)). It is an integral part of the Systems Engineering field which is a prerequisite to every successful development project. The increasing complexity of systems and the fact that requirements should take every stage of the system or product life cycle into account poses ever-increasing challenges to RE discipline ([Wiesner et al., 2015](#)). The following sections give an overview of the most common requirements prioritisation methods, explore the effects of the requirements interdependency and investigate the benefits of requirements traceability.

2.12 Requirements Prioritisation

There is a general agreement amongst many organisations as well as academia that the prioritisation of requirements is necessary because of their varying importance ([Achimugu et al., 2014](#)). It is believed that requirements prioritisation will improve the decision making and ensure an efficient requirements engineering process in product development. The decisions made, however, should be based on rational and quantitative data. In software engineering, for example, requirements are implemented with respect to available resources, cost and quality, and delivery time ([Barney et al., 2008](#); [Finkelstein et al., 2009](#)). Therefore, requirements prioritisation can help in planning software releases, budget control and scheduling ([Achimugu et al., 2014](#)) as well as in selecting only value-added requirements from the considered requirements set ([Babar et al., 2015](#)). Many different methods are used to prioritise the requirements. The recent paper by [Achimugu et al. \(2014\)](#) identified 49 prioritisation techniques. The following sections give a brief description of the most prominent prioritisation methods.

2.12.1 Analytical Hierarchy Process

Analytical Hierarchy Process (AHP) is the most widely used method for requirements prioritisation in academic research. It is a statistical assessment technique proposed by [Saaty \(1980\)](#). The AHP method was originally developed to help solve the problems in the areas of social science, economics and management. The method was adopted in the requirements engineering area because of its ability to set priorities and allocate

recourses. The underlying concept of the AHP is based on the pairwise comparison paradigm. Using this method in requirements prioritisation, the aim is to compare all unique pairs of requirements in order to determine the priority of each requirement and quantify its extent. For n number of requirements, $n \times (n - 1)/2$ pairwise comparisons are required. The major reason for AHP being prized as a promising technique for requirements prioritisation is due to the fact that the method possesses the ability to compute consistency ratios across the requirements, therefore providing reliable prioritization results (Karlsson et al., 1998). The main bottleneck, however, is its scalability. If the number of requirements is increased by a factor of 2, the time and effort required to do the pairwise comparisons increases by a factor of 4. This might be the main reason why this method is not widely used in industry.

2.12.2 Planning Game

Planning Game is one of the elements of the Extreme Programming methodology. The principle of this method is based on customers' story telling that captures their needs (Bergin, 2001; Babar et al., 2015). This method is mostly used in software development. Requirements are represented as short stories, which are composed of the few sentences that reflect what the Customer wants. Developers, on the other hand, estimate how much effort is required to build each story. Customers then choose which of these stories to implement based on time and effort estimated by the developers. There are two major parts that characterise this process: release planning and iteration planning. During the release planning part, the decisions are made on the number and complexity of the requirements and when they will be implemented. The iteration planning part is attributed to developers only. During this process their activities and tasks are planned in accordance with the customers' needs (i.e. stories).

2.12.3 Binary Search Tree

Binary Search Tree (BST) is a method that is used to prioritise requirements. It is particularly useful when prioritising large number of requirements. This is possible because the average computational complexity for binary search tree is $\mathcal{O}(n \log n)$ (Karlsson et al., 1998). The BST is a special case of the binary tree method where the nodes are considered as elements of a set. During the prioritisation process all the requirements that are of lower priority than the node priority are placed in the left subtree, and all the requirements that are of higher priority than the node priority are placed in the right subtree. The underlying principle of BST is choosing one requirement to be a top node and then selecting another from the unsorted subtrees for comparison. If the requirement is lower than the top node, then it is compared with the node's left child subtree, if it is higher – with the right child subtree. The process is repeated until all the requirements are ordered.

2.12.4 Cumulative Voting

Cumulative Voting (CV) or a Hundred-Point Method, described by [Leffingwell and Widrig \(2003\)](#), is based on the voting scheme where each stakeholder is given a constant amount of imaginary units that can be used to vote in favour of the most important requirements ([Chatzipetrou et al., 2010](#)). The priority of the requirement in relation to other requirements is defined by the relative preferences of the stakeholder. The stakeholder has the freedom to distribute the points in any way he or she wants. For example, the whole amount of points can be placed on only one requirement; it can be distributed in equal proportions; or distributed according to the dominating importance of the requirements. CV has been proposed as an alternative to the Analytical Hierarchy Process ([Chatzipetrou et al., 2010](#)).

2.12.5 Cost-Value Approach

The Cost-value Approach is based on the AHP method. The focus of this prioritisation method is only on the relative value of the requirement and its implementation cost. The requirements are prioritised according to their value-cost relationship. [Karlsson and Ryan \(1997\)](#) argued that stakeholder satisfaction is important. Three main factors in stakeholder satisfaction were pointed out, namely quality, cost, and delivery. The aim is to maximize the quality, minimize the cost and shorten the delivery time as much as possible. The authors claim that “*prioritizing based on relative rather than absolute assignments is faster, more accurate, and more trustworthy*” ([Karlsson and Ryan, 1997](#): p. 68).

2.13 Requirements Interdependency

In the product development of complex systems all the development stages are highly related. They cannot be treated in isolation. The same applies to individual design stages. The activities carried out and decisions made at every stage are connected. Requirements Engineering is the first stage in the product development life cycle that specifically deals with requirements elicitation, specification and management. Although each requirement is written as a single statement, it does not stand on its own and is rather related to other requirements affecting each other in a complex manner as a result ([Carlshamre et al., 2001](#); [Regnell et al., 2001](#)). In fact, it has been shown in the study by [Carlshamre et al. \(2001\)](#) that only few requirements are singular and roughly 20% of the requirements are responsible for 75% of the interdependencies. These, in turn, affect the number of development activities and decisions made during the product development process in areas such as requirements design and implementation ([Robinson et al., 2003](#)), requirements reuse ([Knethen et al., 2002](#)), change management ([Knethen and Grund, 2003](#)), release planning ([Carlshamre et al., 2001](#)) and testing ([Dahlstedt and Persson, 2003](#)), just to name a few. For example, the change made to one requirement can

impact not only other requirements but can cascade down to later stages of the product development affecting the activities and decisions in these stages. The consequences of identifying requirements interdependencies in later phases of product development can lead to significantly increased project costs (Dahlstedt and Persson, 2005; Walden et al., 2015).

2.14 Requirements Traceability

Requirement traceability can be defined as “*the method of finding the origin of each requirement both in forward and backward direction and to analyse changes that were made to this requirement*” (Khursheed and Suaib, 2015: p. 109). Generally speaking, requirements traceability captures relational information of the requirements which allows to identify the impact of a proposed change. Additionally, the relational information of the requirements can be used as a basis for decision making and design activities in the product development process. The major advantages of requirements traceability are:

- requirements traceability lowers the risk and maintains consistency (Khursheed and Suaib, 2015),
- supports the reasoning of why the system or artefact has been created, modified and evolved (Ramesh and Jarke, 2001),
- it also provides the means to track and verify the requirements (Bouillon et al., 2013).

There is also a general consensus in the academic literature that incorporating requirements traceability in the product development process improves product quality, reduces the product development time and lowers the life cycle costs of the system (Ramesh and Jarke, 2001; Winkler and von Pilgrim, 2010; Bouillon et al., 2013; Khursheed and Suaib, 2015). On the other hand, neglecting traceability can lead to defects thus poor system quality and increased project time and cost.

Summary

This section has provided an overview of the most common requirements prioritisation methods, explored the effects of the requirements interdependency and investigated the benefits of requirements traceability. Roughly 20% of the requirements are responsible for 75% of the interdependencies that affect the number of development activities and decisions made during the product development process. The change made to one requirement can impact not only other requirements but can cascade down to the later stages of the product development affecting the activities and decisions in these stages. The three major advantages of requirements traceability have been identified. First, requirements traceability lowers the risk and maintains consistency. Second, supports the

reasoning of why the system or artefact has been created, modified and evolved. Third, it provides the means to track and verify the requirements. Incorporating requirements traceability in the product development process improves product quality, reduces the product development time and lowers the life cycle costs of the system.

Cost Estimation

Cost estimation can be defined as a forecast of the probable worth of an activity or resource based on the scope of the estimate and its uncertainty level present at the time (Foussier, 2006). Cost management plays a big role in defining the competitive edge and is an important characteristic of performance for the companies and organisations. To achieve higher profits one can either reduce the costs or increase the price of the product. The former approach is often more desirable for the company since it can have direct control over their costs. The price, however, is often controlled by supply and demand laws of the market.

Forecasting is not an exact science (Foussier, 2006). Hence, the precision of the estimate will depend on the level of detail and uncertainty present at the time when calculating the estimate. The ‘good’ estimate is not necessarily the precise estimate but rather the credible and reliable estimate (Trendowicz, 2013). The cost engineers or estimators face a challenge of providing good a estimate with sometimes limited data or high uncertainty. Because underestimation might affect the profitability, quality and credibility of the company. Overestimation on the other hand might lead to a loss of business or reduced profit margins. According to the literature, little attention has been given by the research community to cost estimation at the early design stages. As mentioned in Section 2.2 and illustrated in Figure 2.6 committed cost accounts for more than 80% of the total product life-cycle cost in the preliminary design stage. However, only 25% of validated design knowledge is available at the early design stage. This confirms the necessity of good estimation techniques and methods that could be used in early design stages and could help cost engineers to provide the credible and reliable estimate (Trendowicz, 2013). In addition, propagating more legacy design and cost knowledge to early design stages should help design engineers to do better concept selection and trade off studies and provide more information for cost engineers for cost estimation activities.

2.15 Cost Estimation Techniques

The primary criteria for choosing the cost estimation technique is mainly based on the company’s or estimator’s experience and the availability of the historic data. In addition, there could be a secondary criteria that might help choosing the cost estimation technique (or a combination of techniques). For example, the purpose of the estimate, the project’s scope, the availability of the resources and the level of risk and uncertainty. The most popular techniques that are used in industry are presented in the following sections.

2.15.1 Parametric

Parametric estimating technique is often used when the information is limited. It relies on the data derived from the similar historic information which is complemented with statistical data. Statistical analysis is performed to establish the correlations between the parameters. The output of the statistical analysis is the equation that defines the relationship between the cost and the system parameter or a set of parameters. For example, the volume of the similar historic part could be used to estimate the cost of the new part, or a set of similar functional cost drivers could be used to establish relationship to cost that would be used for new cost estimate. However, a substantial amount of work needs to be done before using the parametric estimation technique. To make the credible and reliable estimate the parameter correlation analysis needs to be done at the lowest level of the system to make sure that it produces sensible results. For example, the system and sub-system parameters need to be broken down to component level parameters; then the correlation established between the parameters needs to be analysed to make sure that it gives a valid output. Because the relationship with one parameter can sometimes give more accurate or more appropriate correlation factor than the set of parameters.

2.15.2 Case Based Reasoning

Case based reasoning otherwise known as the analogy method is used to estimate cost when the new product is very similar to the legacy products. It is used under the assumption that similar products have comparable costs. However, such an assumption is not always true. Factors such as manufacturing region, labour rates or development effort can be different for the same or similar new product. Nonetheless, this method can provide a quick and reliable baseline estimate from the legacy database assuming that product design, manufacture and economics are similar. The analogy method can also be used to identify the differences between the new and legacy products (Ling et al., 2006). By knowing the differences solutions can be proposed and cost adjustments can be made. The case based reasoning method to estimate cost is often used in early design stages when detailed information and trade knowledge about the product is limited. This method can also be used to estimate cost when the product is standardised. For example, the same bearing can be used across different engine models.

2.15.3 Bottom-Up

The bottom-up technique to estimate cost is often used when the product definition is mature and has a high level of detail. When all the drawings and specifications are finalised the material, overheads, direct and indirect costs can be calculated. In addition, when the manufacturing route is defined the direct and indirect labour costs can also be calculated based on the supply chain conditions. The final cost of the product or a

project is the sum of all the costs. The main advantage of the bottom-up technique is that it provides an accurate cost estimate. The main disadvantage, however, is that it is very time consuming and requires a lot of resources.

2.15.4 Expert Judgement

According to the Oxford dictionary an expert *‘is a person who is very knowledgeable about or skilful in a particular area’* ¹. An expert, especially one working for a long time in the company, could have a fairly good knowledge about the cost of the product. Therefore, the expert judgement is often used in early product development phases or when other techniques or data are not available to provide a Rough Order of Magnitude (ROM) estimate. In addition, this estimation method is quick and requires little resource. Although expert judgement has been widely used and in some cases provided more accurate estimates than conventional techniques one could argue that it is subjective and could be open to bias (Rush and Roy, 2001). The fact that it is based on a personal opinion introduces the element of subjectivity. Furthermore, political intentions, limited resources, time pressure and personal experience can lead to biased decisions resulting in a biased estimate. According to Rush and Roy (2001), there are further limitations of this method, such as lack of consistency, structure, the reasoning is often known only to an expert, knowledge loss if expert leaves the company, hard to audit, difficult to reuse the estimate and it is often difficult to validate the estimate.

2.15.5 Feature Based Costing

Feature based costing (FBC) uses feature characteristics and parameters to calculate product cost. The growth of 3D CAD/CAM modelling tools and its improved capability expanded the use cases of the tool (Roy et al., 2005). The captured feature information can now be used not only in design and manufacturing domain, but also in cost domain. This enabled the development of the FBC method (Caprace and Rigo, 2012). The general consensus is that each product feature has cost implications. Therefore, including or excluding the feature will affect the whole system cost as well as the product lifecycle cost. The main advantages of using this method are: it links the design choices to cost; the method does not require any previous knowledge or data; it can provide a very accurate cost estimate (given that the product model is detailed) (Caprace and Rigo, 2012). The main limitations on the other hand are: the method is time-consuming; it cannot be used in early design stages to estimate the cost of the product due to the fact that it requires the detailed definition of the product; the problem of defining the boundaries of the feature as well as the feature itself in the design and manufacturing domains. Roy et al. (2005) argues that failing to define the feature boundary can affect the cost estimate, because in cases where one feature is comprised of several other

¹<https://en.oxforddictionaries.com/definition/expert>

features there is a danger to double account some of the cost elements (e.g. material costs).

2.15.6 Activity Based Costing

Activity Based Costing (ABC) is a method that allocates the costs to the activities, performed by the company, that are the real cause of the overhead. The product cost is then calculated based on the activities that the product is actually demanding. For example, the cost of the resources used in each of these activities is calculated. Then, the cost of each of these activities will be allocated only to the products that demanded the activity (Skoda et al., 2014). This method, however, requires knowledge of the process to determine the distribution of costs and relies on accurate data (Jurek et al., 2012). The ABC is ideal for identifying high costs of the manufacturing processes and for production optimization, but not useful in early design stages where data is not mature.

Summary

This section has introduced the core techniques and applications in the field of cost estimation. Cost estimation and management plays a big role in defining the competitive edge and is an important characteristic of performance for the companies and organisations. The cost engineers or estimators face a challenge of providing good estimates with sometimes limited data or high uncertainty. On the one hand, underestimation might affect the profitability, quality and credibility of the company, overestimation on the other hand might lead to a loss of business or reduced profit margins. Propagating more legacy design and cost knowledge to early design stages should help design engineers to do better concept selection and trade off studies and provide more information for cost engineers to do the estimation activities.

Chapter Summary

The preceding chapter has endeavoured to provide the reader with an understanding of the four major domains that are related to the current research. The four domains Knowledge Management, Requirements Engineering, Design and Cost domains are interlinked in the product development process. The overview of the research that has been done in those fields is essential in order to help identify the gaps and answer the research questions set out in section 1.3.

Particular emphasis has been given to the field of Knowledge Management and the tools used to capture the design knowledge, predominantly in the early design stages. The argument has been put forward that re-using knowledge will increase the preliminary design knowledge, reduce or even eliminate the duplication of data across systems and reduce the product life-cycle cost as a result. The recent empirical study on design

rationale capture, conducted by [Van Schaik \(2013\)](#), revealed that very little rationale was captured during the detailed design phase and concluded that lack of structured methods on how to link rationale to part geometry appears to be the main limiting factor for the low design rationale capture during geometry design.

Design domain, the second area of interest, has touched on the principles of geometry representation, explored the geometry modelling methods and mechanical engineering design support systems. The current CAD tools capture the geometric relations and geometric design intent but fail to model the actual decision structure or rationale for captured relations. These systems tend to contain the information that describes *what* the product is but not the information of *why* the product was designed in a particular way. The majority of the feature recognition techniques are developed for specific application domains. For example, machining, castings, forging or mould creation domains. A generic algorithm that would provide a fully automated feature recognition process for every manufacturing process still does not exist ([Babic et al., 2008](#)).

Requirements Engineering, the third domain of interest, has presented the concept of requirements prioritisation, interdependency and traceability and has given a brief overview of the most prominent prioritisation methods.

The study by [Carlshamre et al. \(2001\)](#) has shown that roughly 20% of the requirements are responsible for 75% of the interdependencies that affect the number of development activities and decisions made during the product development process. Requirements interdependencies identified in the later phases of product development can significantly increase the project costs.

Requirements traceability captures relational information of the requirements which allows to identify the impact of a proposed change and provides the means to track and verify the requirements. There is a general consensus in the academic literature that requirements traceability adds value to the product development process improving the product quality, reducing the development time and life cycle costs of the system.

The goal of the fourth and final section has been to introduce the core techniques of Cost estimation and emphasise the fact that cost is an important characteristic of performance for the companies and organisations. The committed cost accounts for more than 80% of the total product life-cycle cost in the preliminary design stage but only 25% of validated design knowledge is available at that stage. Therefore, propagating more legacy design and cost knowledge to early design stages should help design engineers to do better concept selection and trade off studies and provide more information for cost engineers to perform the cost analysis and the cost estimation activities.

Chapter 3

Critical Literature Review and Research Focus

3.1 Challenges in Design Rationale Capture

Over the last three decades researchers have proposed and developed a number of tools and methodologies to capture and represent the design knowledge and rationale for future re-use ([Kunz and Rittel, 1970](#); [MacLean et al., 1989](#); [Lee and Lai, 1991](#); [Conklin and Yakemovic, 1991](#); [McCall, 1991](#); [Chen, 1991](#); [Kuffner and Ullman, 1991](#); [Nagy et al., 1992](#); [Myers et al., 2000](#); [Ishino and Jin, 2002](#); [Ahmed, 2005](#); [Auricchio and Bracewell, 2013](#); [Van Schaik, 2013](#)).

A number of developed information representation models were aimed at providing a structured format to represent and document the product development process. Design rationale was captured as a by-product of this approach. These systems attempted to capture the decision making process and transform it into a graph-based node-link structure where design problems (Issues) are emphasised and alternative solutions (Proposals) with the corresponding design rationale (Arguments) are proposed. The fact that design rationale systems can support several goals, such as improving knowledge re-use and design activities, reducing the product life-cycle cost or eliminating the duplication of data across systems are emphasised in the majority of the literature. However, the goals of the system will not be achieved if designers are not using the system. The possible reasons for limited use of the proposed design rationale systems in industry ([Regli et al., 2000](#)) are discussed in the following sections.

3.1.1 Prescriptive and Descriptive Approach to Design Rationale Capture

According to the literature on knowledge re-use, the approach to design rationale capture can be divided into two categories: *descriptive* and *prescriptive* ([Wiegeraad, 1999](#); [Dutoit et al., 2006](#)). A *prescriptive* approach is defined as a method that prescribes a format in which design decision making must take place. For example, the IBIS, PHI,

DRL and QOC methods are the prescriptive approaches. The aim of these methods is to improve the decision making process. This is achieved by using an argumentation-based methodology, which supports the designers in an evaluation of alternative solutions as well as helping them decompose a design problem into sub-problems. Conversely, a *descriptive* approach is not aimed at improving the decision process itself, but rather to capture and document the design decisions for later re-use. For example, the DHT method can be classified as a descriptive approach, because the design decisions can be captured shortly after the actual decision making takes place (i.e. by video-taping the designers decision making process and then translating into a detailed design history) (Chen, 1991; Nagy et al., 1992). The following discussion on the prescriptive and descriptive approaches is based on the description of these approaches presented in Wiegeraad (1999), Dutoit et al. (2006) and Chan (2007).

3.1.2 Advantages and Disadvantages of Prescriptive and Descriptive Approaches

Most approaches to design rationale systems are prescriptive (Wiegeraad, 1999). The major advantage of a prescriptive approach is that the exploration of design problems is improved and the design rationale is inherited as a by-product of this exploration process. In addition, the prescriptive use of design rationale representations gives better insight into the problem and makes the problem more explicit. However, the major drawback of the prescriptive design rationale methods is that they pose a prohibitive overhead on the designers' time for the problems, solutions and arguments to be articulated and structured. This also places an extra burden upon the designer's mind. In contrast, the major advantage of descriptive design rationale methods is that they do not disrupt the normal design process and can be used in any environment. Moreover, the specific style of designing is also left unaffected. Since the design rationale is documented after the decision making process, one of the requirements when using descriptive design rationale methods is to capture the design decisions shortly after the actual decision making takes place. This is because the longer the time interval between the decision and its documentation, the harder it will be to translate the recorded data into design history. This is primarily for three reasons. First, the amount of accumulated data might get too large, making it harder to track and link the raised problems with proposed or rejected alternatives. Second, some important aspects in the decision making process might be forgotten due to a short human memory. Third, due to the unstructured nature of recorded data problems raised, corresponding arguments or decisions made might not be identified. Another drawback of the descriptive approach is that the translated design rationale from the recorded raw data is typically inconsistent and largely depends on the skill and experience of the translator (Chan, 2007). In addition, the decision has

to be made on what information to capture and who is responsible for capturing this information.

3.1.3 Characteristics of Prescriptive and Descriptive Approaches

Prescriptive and descriptive approaches can be characterised by the following aspects: “when to capture” – during or after the decision; “what to capture”; “whose decision to capture” – groups or individuals; “who does the capturing” – the designer himself or others; and “from where to capture” – from people, processes or from product. The distinctive characteristics of the two approaches in the light of these aspects are discussed further below.

3.1.3.1 When

The “when to capture” aspect emphasises the underlying differences between the two approaches. Design rationale can be captured either during or after the decision making process. As discussed in the previous sections, design rationale in the prescriptive approach is captured during the decision making process and is already documented in a structured format, whereas in a descriptive approach it is captured after and therefore the recorded data has to be translated first and only then documented in a structured format.

3.1.3.2 What

In the prescriptive approach the “when to capture” determines the “what to capture”. In other words, once the problem (Issue) is raised, the rationale (Argument) is captured which then supports or rejects the alternative solutions (Proposal). On the other hand, “when to capture” is initiated once the Issue is raised. However, if the problem is not raised in the first place, the corresponding rationale will not be captured. In a descriptive approach, the “what to capture” depends on the observers ability to detect the problems raised and to identify the corresponding rationale from the raw data which was recorded in a non-structured format. For example, the raised problem might not be detected and the corresponding rationale might not be identified by the observer, even though they are recorded. In addition, a designer’s decision might be recorded, but the reasons might not be explicitly stated as he/she might have thought that such a decision was obvious. Therefore, the rationale behind the decision will not be documented.

Since the rationale in the descriptive approach is documented after the decision making process, the “when to capture” is always known. However, the “what to capture” is not always known.

3.1.3.3 Whose

There are three choices to “whose decision to capture”: individual decision, group decision or both. The group decisions are often higher level decisions related to conceptual or embodiment design (Chan, 2007). In detailed design individual decisions prevail and are left to individual designers. During group discussions, decisions made and reasons behind the decisions are expressed in words therefore can be easier identified and captured by the observer. On the contrary, decisions made by individual designers and reasons behind the decisions reside in their minds and have yet to be communicated (verbally or literally) in order to be documented by the observer. Both, group decisions and individual decisions are equally important as they are interrelated and help clarifying the choices made throughout the design process. Therefore, both individual decisions and group decisions are captured in prescriptive and in descriptive approaches.

3.1.3.4 Who

In prescriptive approaches the design rationale is captured by the designer. The major drawback of the prescriptive approach is that it disrupts the normal design process. Furthermore, the designers have to learn how to present their decision making process in terms of the semiformal node-link structures. It is not always easy for designers to express their thinking in terms of Issues, Proposals and Arguments, especially under time constraints. Schedule pressure might lead to capturing only the bare minimum of the design process or even lead to failure in capturing any information (Chan, 2007).

While in prescriptive approaches it is clear who captures the design rationale, in descriptive approaches, however, the capturing can be done by all designers, by several specific designers, by the observant, or a combination of them. Therefore, it is important to define who does the capturing of the design rationale. As discussed in previous sections, a descriptive approach is beneficial when the decision making process or the decisions made during group discussions are recorded first and then translated and structured into useful design history. Another benefit is when the observant does the capturing because such an approach does not disrupt the normal design process. However, the quality of captured information depends on how well the observant interpreted the records, and translated and structured them into useful formats (Chan, 2007).

3.1.3.5 From Where

Cross (2006) identified three sources of knowledge: *people*, *processes* and *products*. Design knowledge resides in everyone involved in the product development process, especially in designers. This source of knowledge is kept in the designer’s head and in his/her notebook and is not readily available to others. Design knowledge residing in *processes* is in the form of drawings, specifications, design definition reports, and user manuals that resulted from product development processes. This form of knowledge was

primarily aimed at providing information required to manufacture, use and maintain the released product. Design knowledge residing in *products* is expressed in the forms, fits and materials which embody design attributes.

The design rationale in the prescriptive approach is a by-product of the decision making process, which is bounded and restricted by a method used (i.e. IBIS, DRL, QOC etc.). Designers capture their decision making process in terms of the semiformal node-link structures where problems or questions are linked to the alternative solutions with the rationale behind them. The fact that knowledge resides in designer's head leads to a conclusion that in the prescriptive approach rationale is captured from *people*. Conversely, the descriptive approach does not impose any restrictions on design and decision activities. The design rationale is captured after the decision has been made, from raw data recorded in the form of videos, meeting notes, drawings, reports, emails or any other data that has been stored in an unstructured manner. Therefore, it can be concluded that in the descriptive approach rationale is captured from *processes*.

3.1.4 Prescriptive or Descriptive

The aspects that characterise the prescriptive and descriptive approaches are summarised in Figure 3.1 and Figure 3.2, respectively.

	When	What	Whose	Who	Where
Prescriptive	During	Known	Individual	Designer	People
					Processes
	After	Unknown	Group	Observer	Product

Figure 3.1: The aspects that characterises the prescriptive approach. Reproduced from [Chan \(2007\)](#).

The prescriptive approach is bounded and restricted by a method used, therefore is too restrictive and inflexible. The descriptive approach, however, does not impose any restrictions on design and decision, yet it is too general. It results in vast amounts of recorded raw data that could be difficult to structure into useful information. As mentioned previously, both approaches have advantages and disadvantages. However, the limited use of the Design Rationale capture tools in industry suggests that the implementation of such approaches in a real design environment is challenging. In order to provide design engineers with the tools that support the creation, exchange, reuse and management of design information, mechanical engineering design support systems have been developed ([Chan, 2007](#)).

The current support systems that are widely used in industry are discussed in Section 2.8.

	When	What	Whose	Who	Where
Descriptive	During	Known	Individual	Designer	People
					Processes
	After	Unknown	Group	Observer	Product

Figure 3.2: The aspects that characterises the descriptive approach. Reproduced from [Chan \(2007\)](#).

3.1.5 The Dilemma of Knowledge Capture Tools

Figure 3.3 summarises the knowledge capture tools currently used in the product design process. As seen in the figure, the majority of current knowledge reuse tools are specific applications with descriptive or prescriptive approaches. Furthermore, the majority of design rationale capture tools are generic prescriptive in nature.

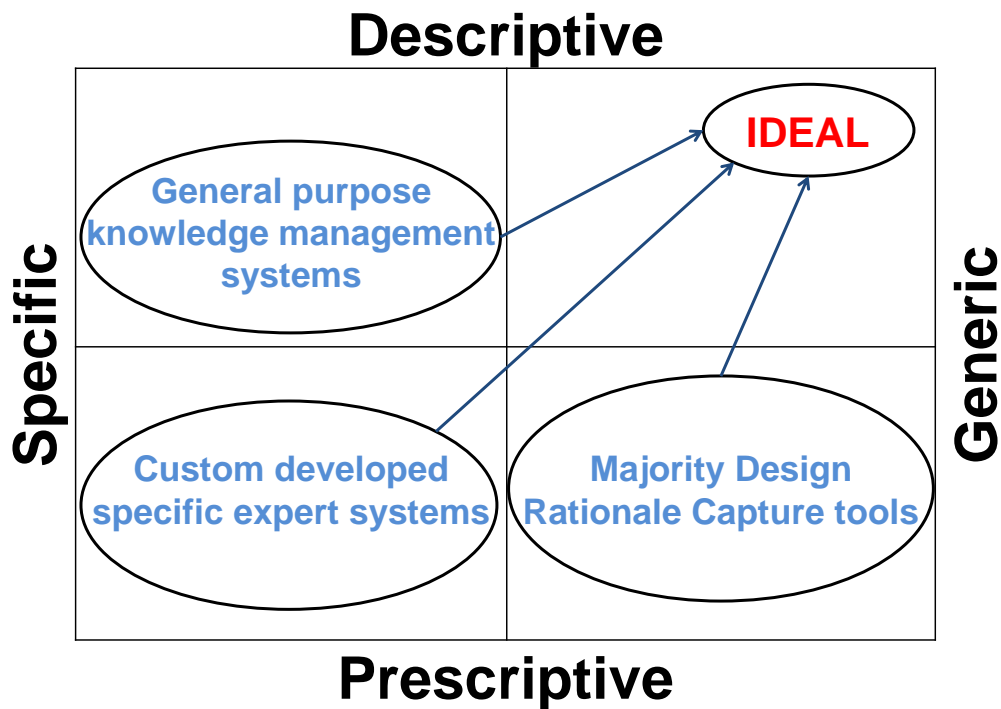


Figure 3.3: The knowledge capture tools currently used in the product design process.

As Figure 3.3 suggests, the relatively ideal approach should be generic descriptive. Generic means that the approach is not restricted to a particular product and is applicable to any design process. Descriptive means that the methodology used to capture the design rationale should not restrict or bound the designers by a prescribed method. Figure 3.4 illustrates the aspects addressed by the descriptive approach and generic design tools.

It is interesting to note, however, that none of the currently used approaches capture the design rationale from product (Figure 3.5). Cross (2006) argued that design knowledge also resides in products and is expressed in the forms, fits and materials which embody design attributes. Research has identified that up to 90% of new designs are based on existing designs (Kim et al., 2007). Therefore, product knowledge of existing designs can be reused in future product designs as a result improving the productivity and efficiency of the design process.

	When	What	Whose	Who	Where
Generic and Descriptive	During	Known	Individual	Designer	People
					Processes
	After	Unknown	Group	Observer	Product

Figure 3.4: The aspects addressed by the descriptive approach and generic design tools. Adapted from Chan (2007).

	When	What	Whose	Who	Where
Prescriptive/ Descriptive and Generic/ Specific	During	Known	Individual	Designer	People
					Processes
	After	Unknown	Group	Observer	Product

Figure 3.5: The aspects addressed by the prescriptive and descriptive approaches to design rationale capture. Adapted from Chan (2007).

3.2 What Knowledge Resides in Product?

Cross (2006) identified three sources of knowledge: people, processes, and products. Reviewing the literature it was found that a lot of work has been done in the knowledge

and design rationale capture fields. A number of information representation models were aimed at providing a structured format to represent and document the product development process. The design rationale was captured as a by-product of this approach. The majority of the tools focused on capturing knowledge mostly in the concept and preliminary design stages. These systems attempted to capture the decision making process and transform it into a graph-based node-link structure where design problems (Issues) were emphasised and alternative solutions (Proposals) with the corresponding design rationale (Arguments) were proposed. As mentioned in Section 2.6, only a few empirical studies with quantitative results on design rationale capture during the design process have been published. The aim of the recent empirical study on design rationale capture during the UAV design process, conducted by Van Schaik (2013), was to measure the quantity of rationale that had been captured during the design stages. The main focus of this study was on the knowledge that resided in product and ‘*how*’ much of it could be captured from that product. The focus of the current study, however, is on ‘*what*’ knowledge can be captured from product. The aim is to investigate what type of data and information is stored within the product, how easy it is to extract and to clarify any data dependencies that could influence decision making during the design process. The following section describes the case study setup and findings.

3.2.1 The Case Study

The objective of this case study was to investigate what type of information resides in the geometry model, how easy it is to understand the design intent from the modelling history of the part and what story it tells about the decisions made. The idea was to investigate the part models of Rolls-Royce that sponsors this research. A variety of Siemens NX CAD models (the primary CAD tool used by Rolls-Royce) were investigated to establish what product data was available, the meaning, or semantics, of the data and whether it was possible to understand the design process.

Six classes of knowledge, namely formal, tacit, compiled, dynamic, process, and product knowledge are discussed in Section 2.2. The latter is the most prevalent in geometry modelling. There are two main sources of information about the geometry of the product; the part model itself and the drawing of the part. The geometric and topological information, material data as well as information about the modelling features that are used to construct the geometry is stored in the part model. In addition, information about the procedural order of the modelling operations is stored in the modelling history tree. The Geometric Dimensioning and Tolerancing (GD&T) data together with the manufacturing information (surface roughness, coating type etc.) is stored in the part drawing. The information types above can be characterised as *explicit* information, because it is explicitly specified in the part model and in the drawing of the part. However, the relational data between the modelling features and geometric entities is not

that obvious. Therefore, this type of data can be characterised as *implicit* information. To illustrate the latter statement the Rear Lock Plate CAD model was chosen as an example case study.

The Rear Lock Plate is part of the Trent High Pressure Turbine sub-system. The primary function of the lock plate is to provide axial retention for the Turbine Blade. The secondary function of the lock plate is to provide a seal for the local Air System. The Rear Lock Plate interfaces with the seal plate, HPT blade, damper and HPT disc (Figure 5.2). The Rear Lock Plate CAD model is shown in Figure 3.6. It illustrates the geometric and topological information as well as information about the modelling features. The GD&T data and the manufacturing information is also explicitly specified in the Rear Lock Plate drawing (Figure 3.7). However, due to the sensitivity of the part drawing has been redacted.

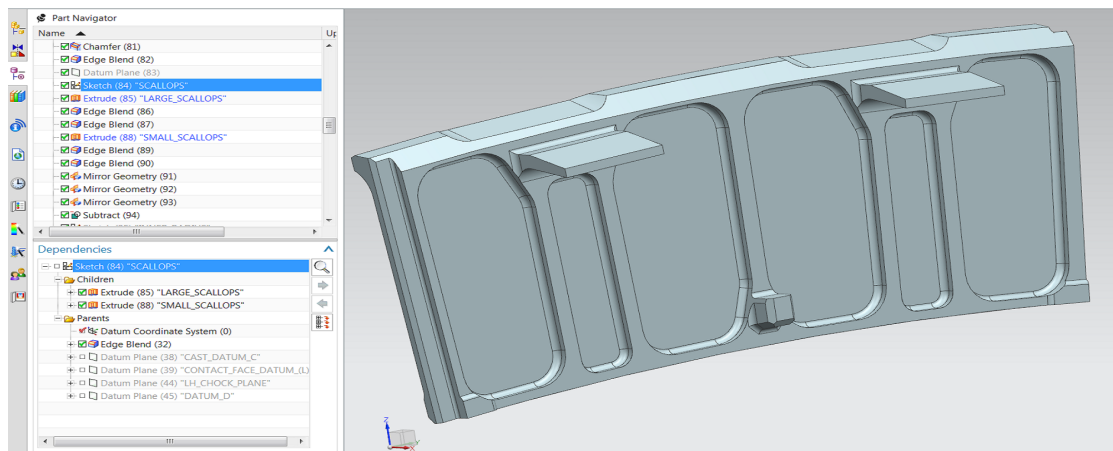


Figure 3.6: The Rear Lock Plate CAD model.

The specified information about the Rear Lock Plate, as discussed above, was expected to be found in the part model and in the drawing of the part. The model, however, lacked semantics. Three shortcomings were identified. Firstly, only a few modelling features had meaningful names assigned to them. Therefore, it was hard to understand which design features had been modelled and what modelling features (in the modelling history tree) were necessary to produce the design feature. Figure 3.6 shows all modelling features listed in the part navigator panel on the left hand side, but only few have meaningful names assigned. In the author's opinion, it is a crucial information because it signifies component's function and design intent. Therefore, having the semantic information specified in the part model would save time during the redesign activities, especially for a new team with no previous knowledge about the part. There will be no need to spend extra time identifying modelling features defining the design feature. Furthermore, specifying the design features that best describe their function will significantly enhance the knowledge about the part, enabling a better understanding about the design intent as a result. The possible reasons for a lack of semantics or textual information might be

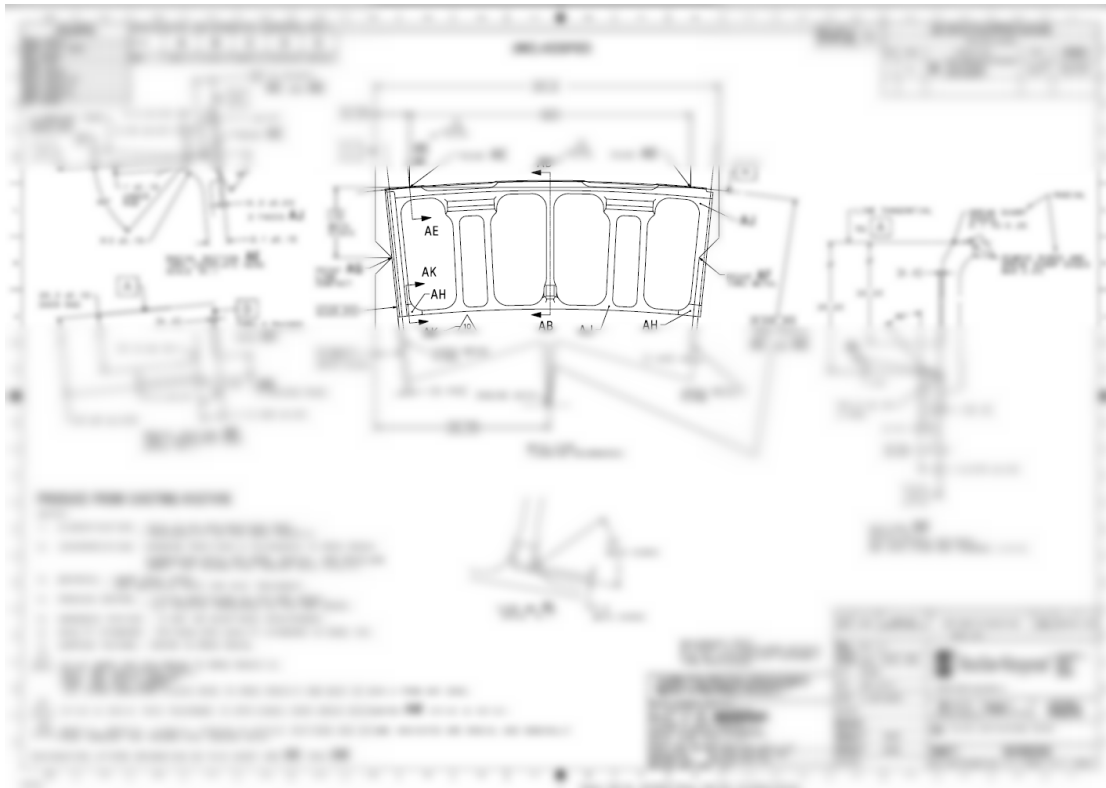


Figure 3.7: The Rear Lock Plate drawing.

due to time pressure or a lack of motivation as identified by [Van Schaik \(2013\)](#). To prove the point, the feature groups were added to the part model to signify the component's function and design intent. An example of the Rear Lock Plate part model with the defined design features is illustrated in Figure 3.8.

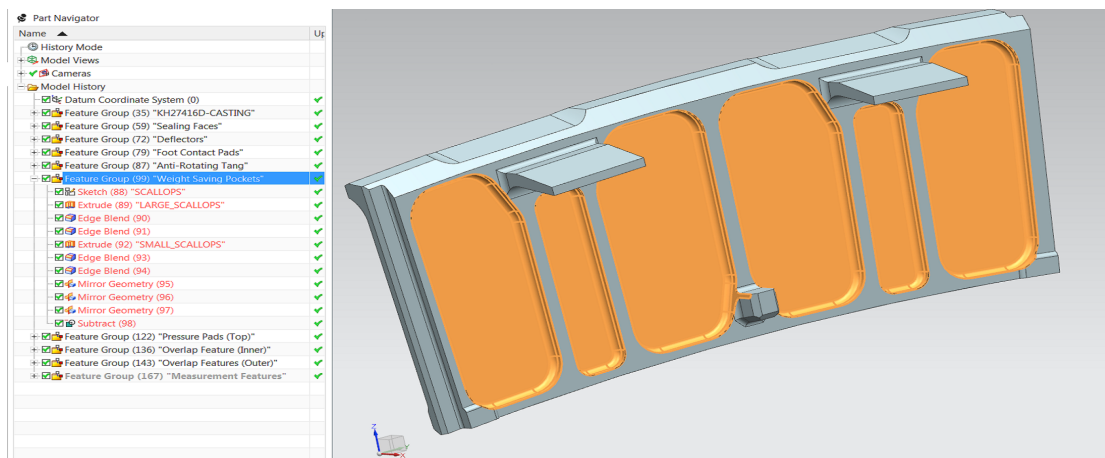


Figure 3.8: The Rear Lock Plate part model with the defined design features.

As seen in Figure 3.8, the design feature '*Weight Saving Pockets*' (marked in orange) is clearly defined by the modelling features under the *Feature Group (99) 'Weight Saving Pockets'* in the part navigator. By grouping the relevant modelling features that define

the design feature and adding semantic meaning, improves the understanding of design and captures the functional knowledge from the part model. Figure 3.9 illustrates the part model with and without the information about the design features.

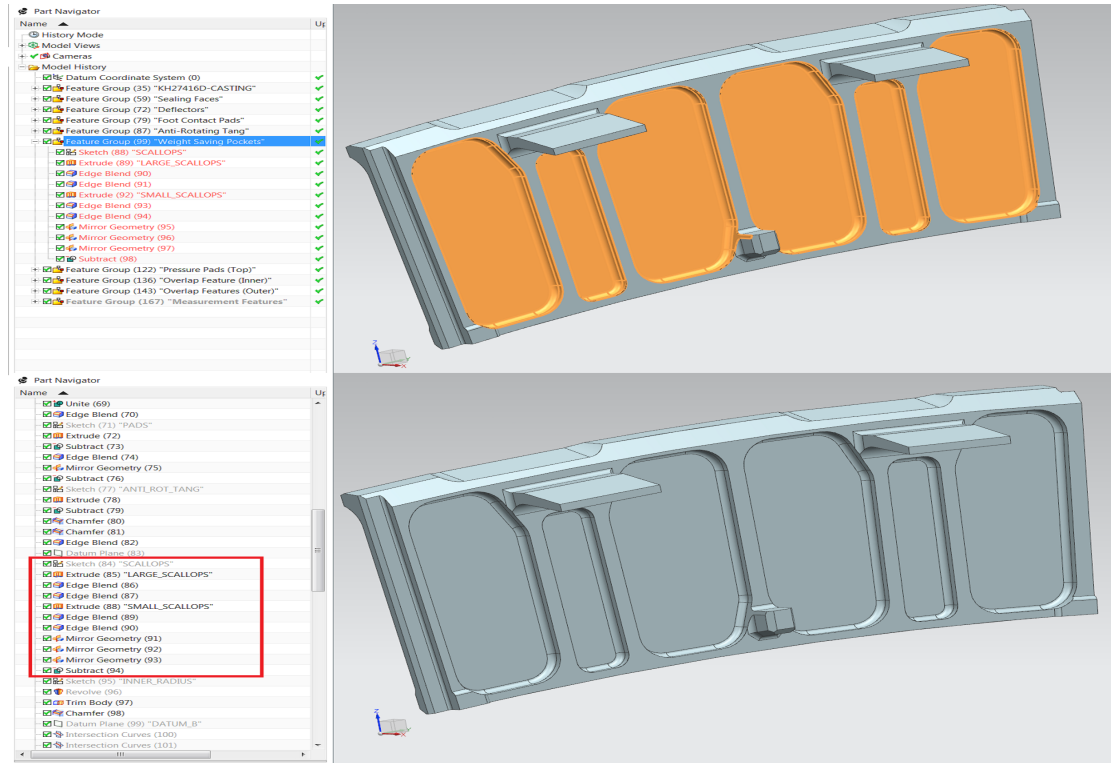


Figure 3.9: Part models with and without the information about the design features.

The second shortcoming was the different perception of features in the design domain and manufacturing domain. It was observed that the design feature '*Weight Saving Pockets*' in the part model was defined as a '*Milling Feature*' in the drawing of the part. Moreover, the geometric data of the design feature was different from the geometric data of the manufacturing feature. For example, the design feature '*Foot Contact Pads*' shown in Figure 3.10, does not even exist in the manufacturing domain because it is a by-product of the milling operation and is classified as the subset of the milling feature. In fact, both design features '*Foot Contact Pads*' and '*Stiffening Ribs*' are the by-products of the milling operation that produces the '*Weight Saving Pockets*' design feature shown in Figure 3.11. The main challenge is to recognise features in the design domain and then transform them into the manufacturing features. As discussed in Section 2.11 and recognised by other researchers, it is a complex problem for which a generic solution has not yet been found.

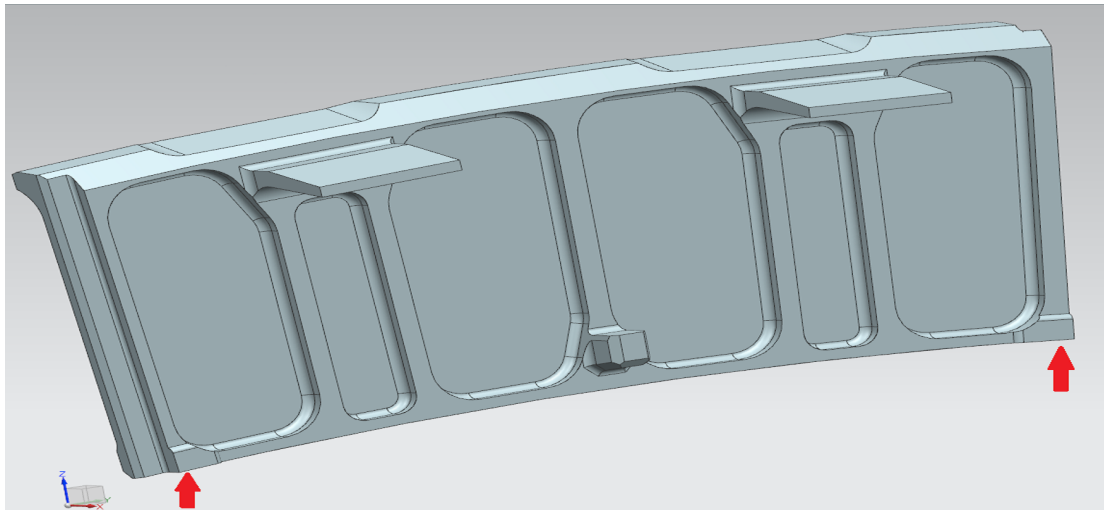


Figure 3.10: Design feature '*Foot Contact Pads*.

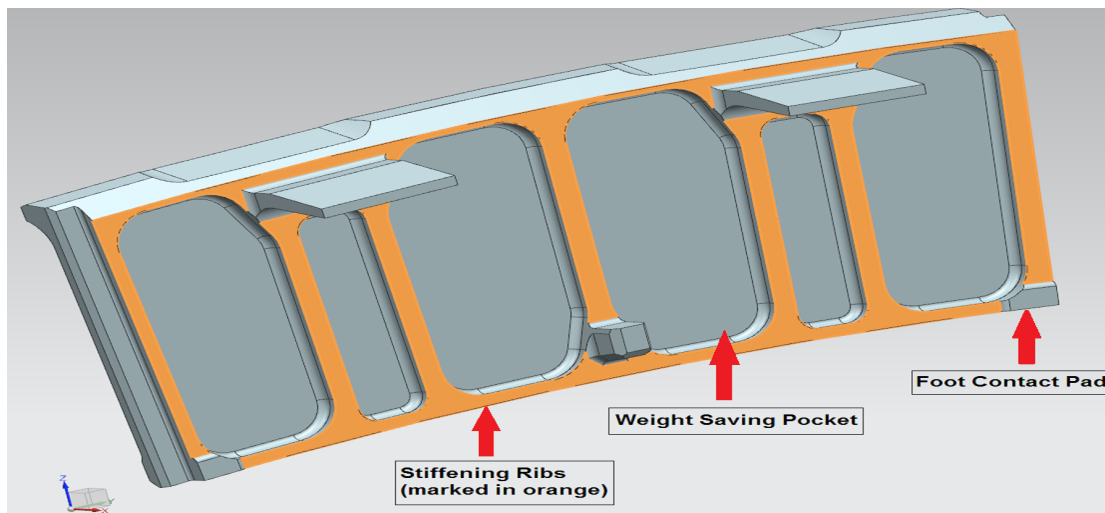


Figure 3.11: Design features: *Foot Contact Pads*, *Stiffening Ribs* and *Weight Saving Pockets*.

Thirdly, implicit relational data between the features and geometric entities was hard to visualise. Although the relational data could be accessed via the Dependency panel shown in Figure 3.12, due to complexity of the geometry it was challenging to visualise the whole dependency map. This became particularly relevant when the part had a very complex geometry, therefore having a visual dependency graph should help when evaluating the component's complexity. To capture the Parent-Child relationships, a script (Appendix A) was written using the NXOpen API library to extract the data. The BOXARR Software tool (Section 4.3.1) was then used to visualise the dependency map. The relational data between the modelling features was extracted for both part models of the rear lock plate, with and without the definition of the design features. Figures 3.13 (with the definition of the design features) and 3.14 (without the definition

of the design features) show the relational dependencies between the modelling features. The purpose of these graphs is to demonstrate the complexity (and not the actual data) and to show that the data in Figure 3.13 is more structured. Design features are grouped under the Design Features Group (refer to the red rectangle in Figure 3.13).

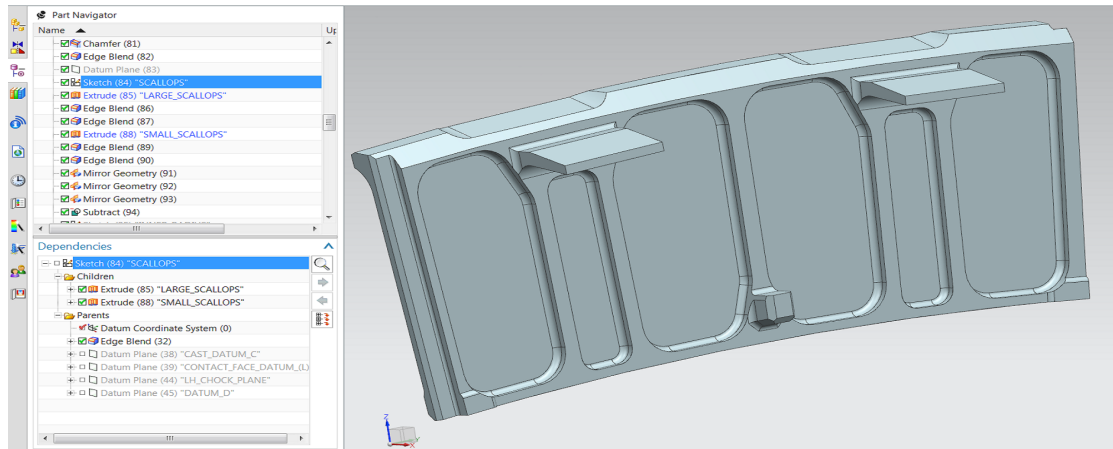


Figure 3.12: Relational data could be accessed via the Dependency panel in part navigator pane.

From both graphs it is evident that dependencies between the modelling features are complex. Therefore, graphing tools like BOXARR are necessary to allow relational dependency analysis within the CAD software to visualise the overall impact of the change when one or more modelling features are changed. It should be noted, however, that the semantics within the part models need to be present in order to make the dependency analysis efficient. For example, suppose a design engineer starts the dependency analysis between the modelling features of the Rear Lock Plate and wants to know the dependency relations of the modelling feature '*RH_OVERLAP_CUTTER*'. He/she filters upstream and downstream connections of the feature that are shown in Figure 3.15. The figure shows two types of graphs (the same data has been used to create graphs in Figures 3.13 and 3.14). The top graph demonstrates the filtered relational data of the modelling feature '*RH_OVERLAP_CUTTER*', but without the defined design features, whereas the bottom graph illustrates the defined design features. Although the modelling feature '*RH_OVERLAP_CUTTER*' in the top graph suggests (i.e. the naming or semantics of the feature) that it is related to the overlap design feature, a designer will not know that it also affects the measurement features. Therefore, the bottom graph provides much more information for dependency analysis leading to a better decision.

It can be concluded that by simply grouping the modelling features in the model that characterises the design feature significantly enhances the product knowledge. This enables designers to make better decisions during the part redesign activities and relational dependency analysis.

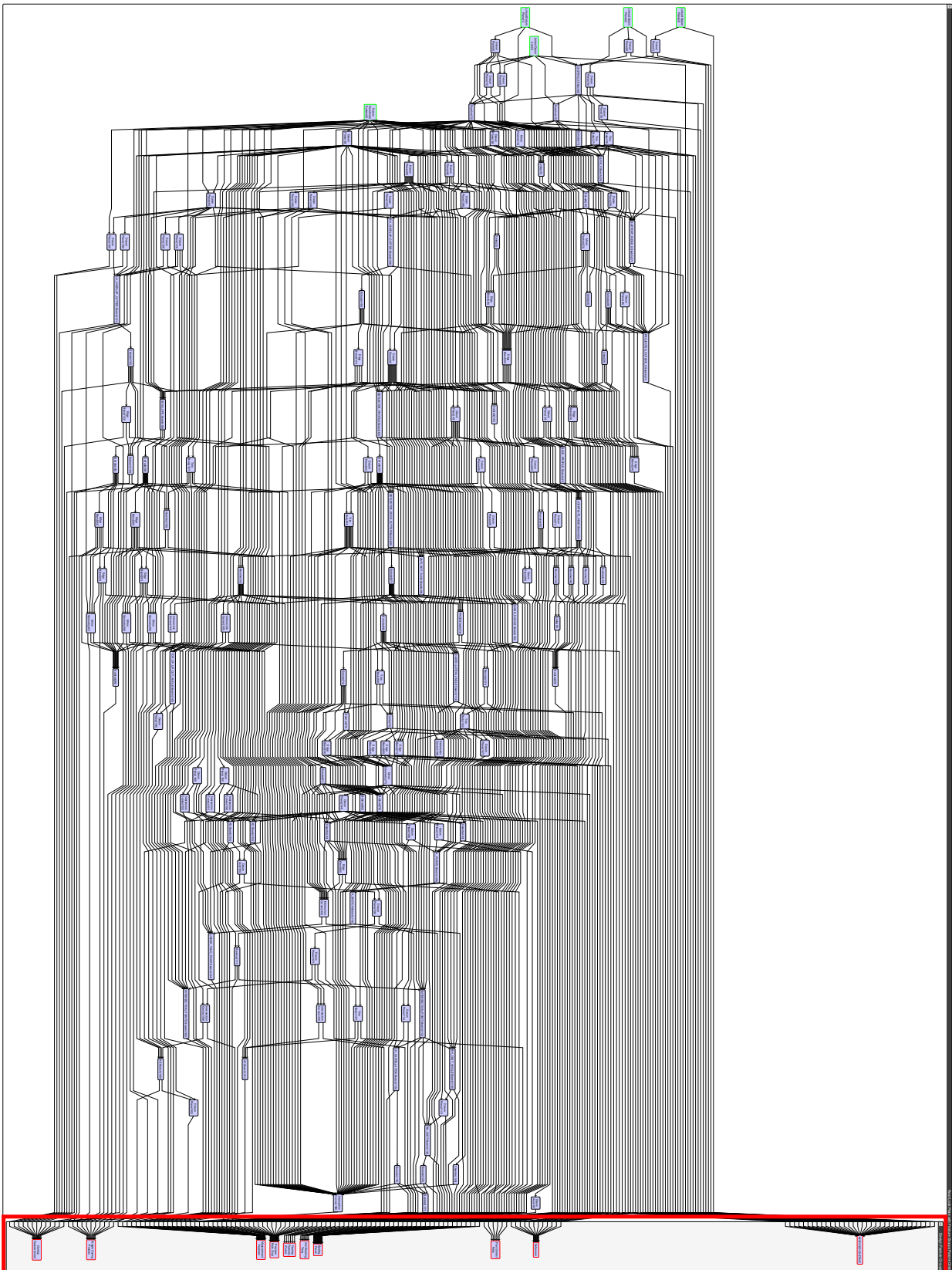


Figure 3.13: Relational dependencies between the modelling features. The design features are defined.

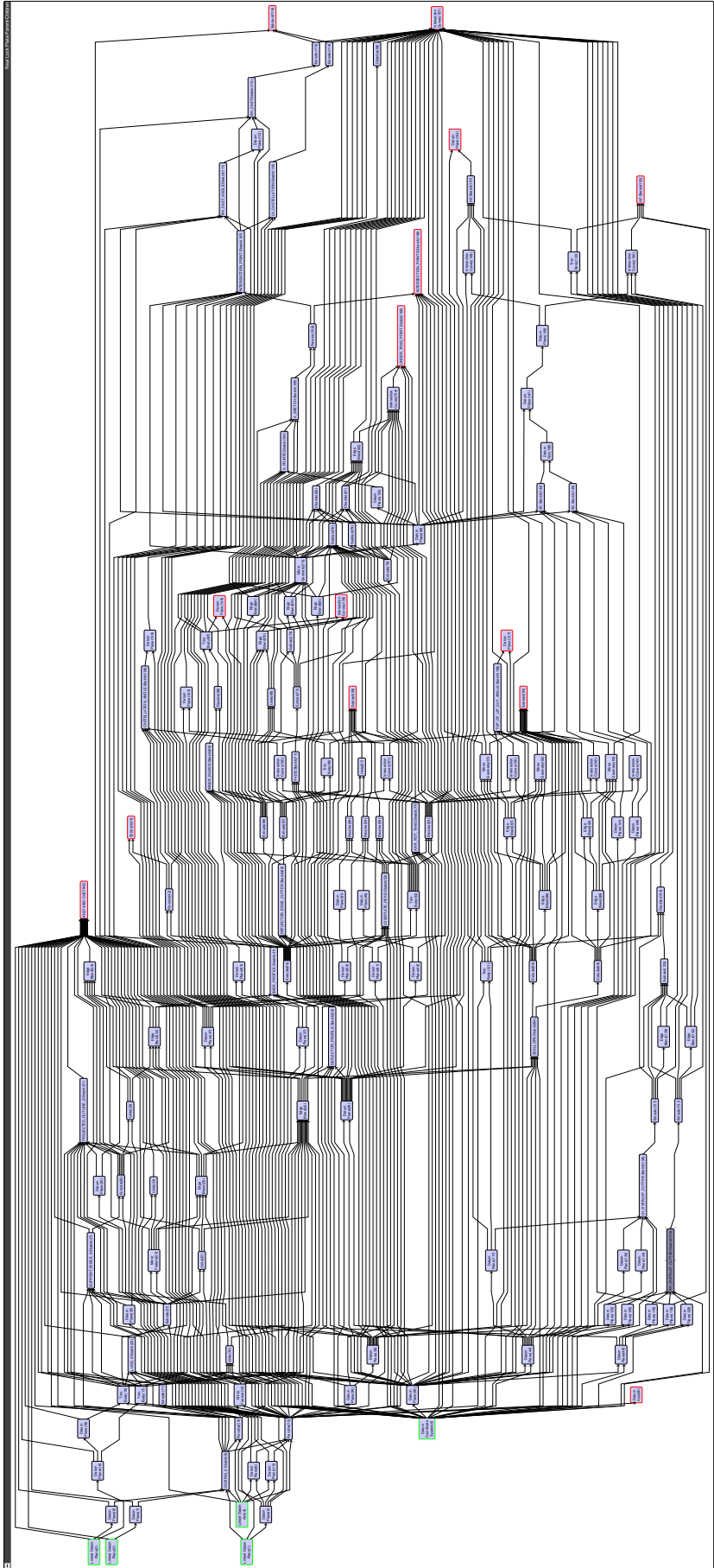


Figure 3.14: Relational dependencies between the modelling features. The design features are not defined.

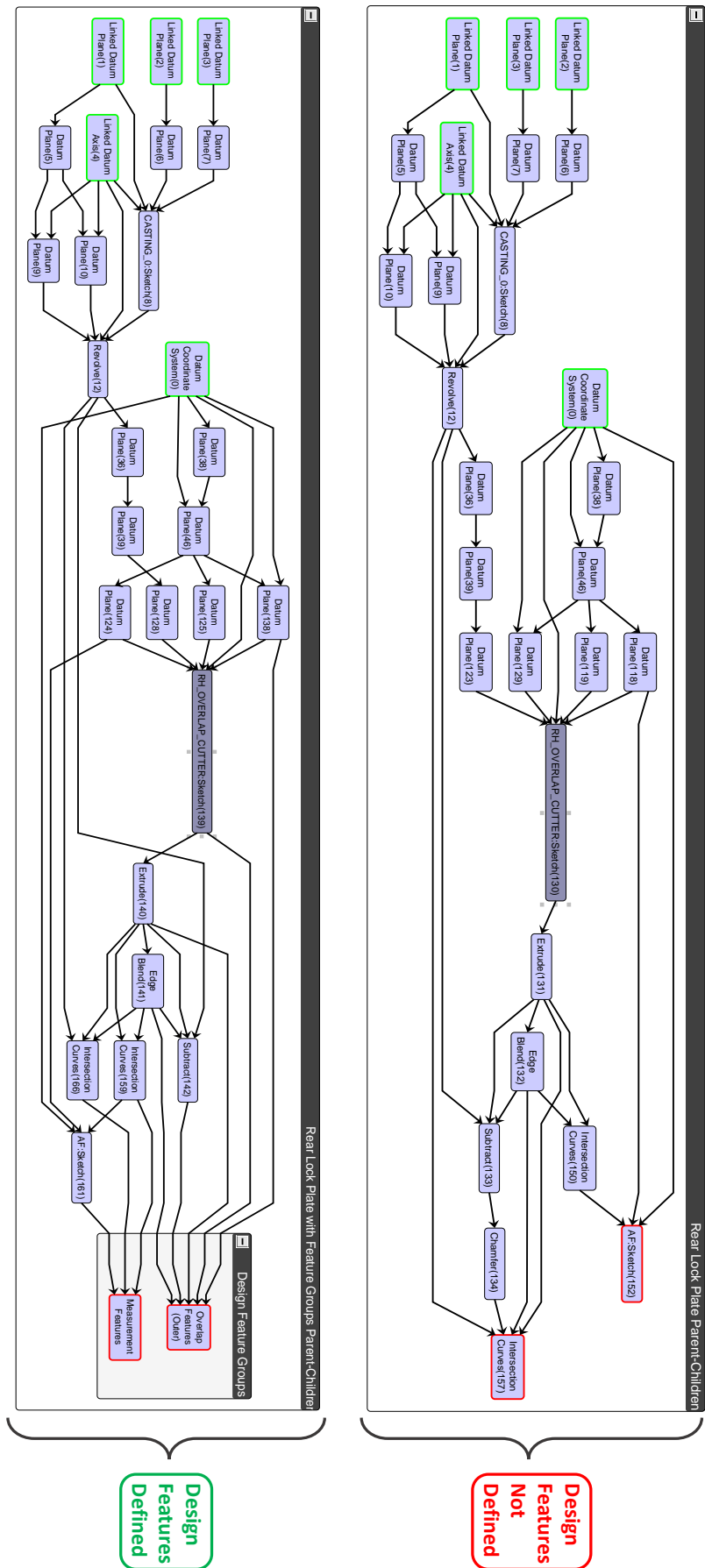


Figure 3.15: Filtered upstream and downstream connections of the 'RH OVERLAP CUTTER' modelling feature.

3.3 Challenges in Requirements Engineering

The product development process is initiated by the customer needs that are translated into the requirements, which state *‘what’* the system or product should be. Requirements engineering is a very important area in the product development process because it not only deals with requirements elicitation and specification, but also manages conflicting preferences and expectations of the stakeholders’ requirements (Karlsson et al., 1997). It is crucial to identify any conflicting requirements at this product development stage. The cost of making changes in the later product development stages have been shown to increase by orders of magnitude (Karlsson et al., 1997; Dahlstedt and Persson, 2005; Walden et al., 2015). As the literature reveals, the interest in requirements engineering research has been increasing in the last two decades. The increasing number of papers in the Scopus database confirms this statement (see Figures 3.16 and 3.17). However, there are still challenges in some areas of requirements engineering research as well as in industry that need to be overcome. The underlying challenges are summarised in the following paragraphs.

Chakrabarti et al. (2004) conducted a study on how requirements are identified and applied during design activities and what the impact of the requirements is on the design process. The authors concluded that requirements identification and application activities span throughout the whole design process; these are necessary in order for the requirements to be adequately fulfilled by the final design. The authors also emphasised the lack of suitable tools in helping to understand the requirements in an informed way in terms of their relational interdependencies as well as their relative significance. The tools should be easy to use and preferably be dynamic in order to satisfy the iterative nature of the design process.

Rios et al. (2007) reported the following findings from the requirements analysis and the interviews with five engineers that were involved in the design process from the aerospace company where the research had been carried out:

- Functional requirements were not explicitly declared.
- The majority of the requirements were specified to meet the airworthiness regulations.
- Many requirements had no means for verification.
- There were no explicit links between the functional requirements and the design parameters.
- Materials selected for the design solution accounted for the majority of constraints.
- The requirements had not been clearly written. Most of the requirements and constraints were not fully clear to the design engineers.

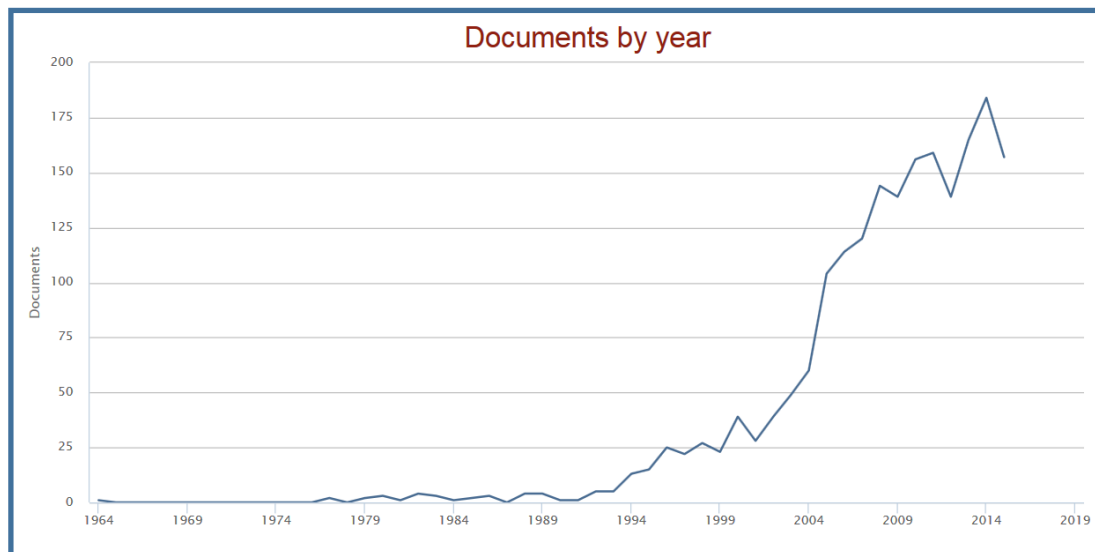


Figure 3.16: Number of documents that have the words '*requirements engineering*' in the title.¹

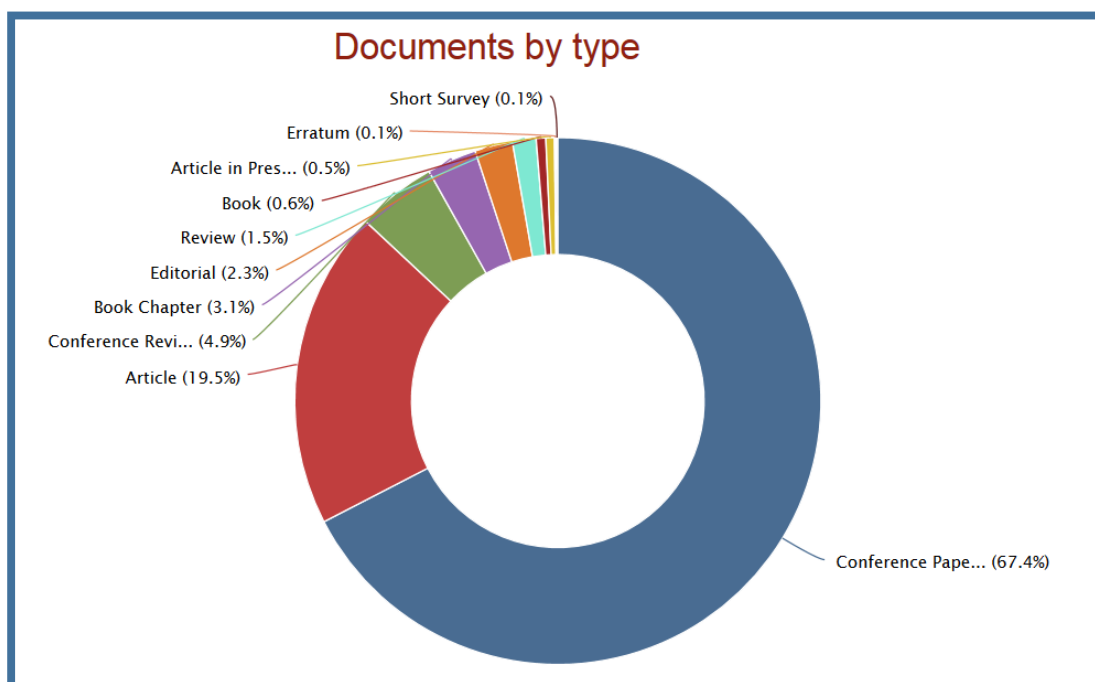


Figure 3.17: The type of documents that have the words '*requirements engineering*' in the title.¹

- The relational dependencies between the requirements had not been explicitly stated.

¹Scopus. <https://www.scopus.com>

[Oduguwa et al. \(2006\)](#) identified the lack of formal methodology to assess the cost of the requirement change. The author argues that the interdependency knowledge of different sub-systems is crucial and necessary for cost impact analysis during the requirements change or redesign activities.

The paper by [Winkler and von Pilgrim \(2010\)](#) stated that requirements traceability practices are not yet mature and that both fundamental and applied research in this area still needed to be done. The authors identified several limiting factors affecting requirements traceability. *Natural* - imprecise and incomplete nature of traces leads to a natural limitation to traceability. One of the potential reasons is that the requirements management software does not provide the capability to trace the requirements. However, the majority of current requirements management tools do have this capability. Thus, the second potential reason is the lack of motivation to establish traces. It is a very similar situation as in design rationale capture ([Van Schaik, 2013](#)). High workload and time pressure leaves little or no time for engineers to establish traces between the requirements. *Technical* - there is no semantically well-defined traceability meta-model which can record traces in an automated way without human intervention. This is a very difficult task due to the fact that the semantics in distinct industries are different. For example, a software engineer is concerned about the code, whereas a design engineer is concerned about the geometry of the component. Therefore, the human-machine approach to establish traces between the requirements seems to be the most efficient at the moment if/when used in industry. *Economical* - the benefit is hard to measure. There is no sound empirical proof that traceability adds value to a project or a company. It is an interesting statement however. In the author's of this research opinion, the main reason why the benefit of the requirements traceability is hard to measure is because the 'time' metric is not often expressed in monetary terms in engineers' day to day activities. For example, engineers are focused on achieving the performance target but they often discount how efficient the process was in achieving the performance target; time wasted to search for relevant information, due to a lack of proper tools, could have been used more productively. Therefore, the argument is that if there were better tools that could support and help improve the engineers day to day activities, time to complete the tasks should decrease leading to an increased productivity, reduced stress, and added value to a project or a company. *Social* - lower quality traces could potentially be recorded due to a lack of motivation. Again, this drawback could be related to the absence of proper requirements traceability tools that allow to establish traces easily.

The empirical study, carried out by [Lehtola and Kauppinen \(2004\)](#), focused on the evaluation of two requirements prioritisation methods from the requirements engineering literature, namely the Wieggers Method and the AHP. The study has been carried out in the industrial environment. The conclusions of the study were that practitioners were unclear how to define the value, penalty, cost or risk factors when prioritising the requirements. In their words ([Lehtola and Kauppinen, 2004: p. 7](#)): "How do I evaluate

a requirement's value to the customer or implementation costs if we do not have any common basis as to what the terms "value" or "cost" mean? What is the information on which I should base my evaluation?". Moreover, the practitioners in industry found it difficult to estimate which number to give to factors: *"What does it mean if I give either number 3 or 5? What is the practical difference between them?"* (Lehtola and Kauppinen, 2004).

In their recent paper, Babar et al. (2015) further pointed out the following problems in the existing software requirements prioritisation techniques:

- The methods are not scalable for a larger number of requirements.
- There is no sufficient automation.
- Most of the techniques are time-consuming.
- The existing methods are complex, therefore difficult to implement.
- The existing techniques deal mostly with small numbers of requirements.

Industrial and academic environments are very different. In industry engineers are constantly battling with high workload and time pressure to meet deadlines. These stresses negatively affect their motivation to use new tools, because often it requires commitment to learn how to use them. Therefore, it is not a surprise that some methodologies, showing a huge potential in academic environment, had not been adopted by the industry. The above notes from academic literature reflect this. In addition, the above remarks show that there is still room for improvement in developing a methodology that is intuitive, easy to use and, most importantly, provides the means for design engineers to make better decisions during the design activities.

3.4 Research Focus

The comprehensive overview of knowledge capture methods and tools as well as challenges has been presented in Chapters 2 and 3, respectively. Knowledge management is a central topic of this research with a secondary focus on the requirements engineering, design and cost estimation fields. In addition to the automated design knowledge capture methodology, which is the main objective, this research also addresses the challenges in the requirements engineering, design and cost estimation fields. The focus of this research will be driven by the following findings from Chapter 3:

- The ideal approach should be generic descriptive. This means that is not restricted to a particular product and is applicable to any design process.

- None of the currently used approaches capture design knowledge from *product*.
- Poor semantic information about the modelling features in the modelling history tree makes it harder to understand the design intent and the function of the feature.
- The perception of features in the design domain and manufacturing domain is different.
- Implicit relational data between the features and geometric entities is hard to visualise.
- The interdependency knowledge of different sub-systems or components is crucial and necessary for cost impact analysis during the requirements change or redesign activities.
- The requirements traceability practices are not yet mature and that both fundamental and applied research in this area still needed to be done.
- It seems that confusion still exists and that practitioners are unclear how to define the value, penalty, cost or risk factors when prioritising the requirements.

The above findings can be translated into the following requirements:

- The knowledge capture approach shall be generic descriptive.
- The knowledge shall be captured from *product* and product development process in an automated way.
- The methodology/tool shall be capable to unify the feature definitions in design domain and manufacturing domain.
- The methodology/tool shall provide the means to visualise relational data across requirements domain, design domain and cost domain.
- The methodology/tool shall be capable to capture and visualise the interdependency between requirements and features.
- The methodology/tool shall be capable to calculate the requirements impact.

Chapter Summary

This chapter has presented the critical literature review and identified the challenges in the fields that are of primary concern in this research. These are Knowledge Management, Design and Requirements Engineering. It has also provided a guide of the areas that this research has focused on.

Two main design rationale or knowledge capture approaches, *descriptive* and *prescriptive*, have been identified. The prescriptive approach is bounded and restricted by a method used, therefore is too restrictive and inflexible. The descriptive approach, however, does not impose any restrictions on design and decision, yet it is too general. The limited use of the Design Rationale capture tools in industry suggests that the implementation of such approaches in a real design environment is challenging. The majority of design rationale capture tools are generic prescriptive in nature. The relatively ideal approach should be generic descriptive. It means that is not restricted to a particular product and is applicable to any design process.

Cross (2006) identified three sources of knowledge: *people*, *processes* and *products*. The findings suggest that in the prescriptive approach knowledge is captured from *people* and in the descriptive approach it is captured from *processes*. However, none of the currently used approaches capture design knowledge from *product*.

The case study *What knowledge resides in product?* has investigated what type of data and information is stored within the product, how easy it is to extract and to clarify any data dependencies that could influence decision making during the design process. The study found that poor semantic information about the modelling features in the modelling history tree makes it harder to understand the design intent and the function of the feature; the perception of features in the design domain and manufacturing domain is different; and implicit relational data between the features and geometric entities has been hard to visualise.

Although the interest in requirements engineering research has been increasing in the last two decades, there are still challenges in some areas of the requirements engineering research as well as in industry that need to be overcome. The interdependency knowledge of different sub-systems is crucial and necessary for cost impact analysis during the requirements change or redesign activities. The requirements traceability practices are not yet mature and that both fundamental and applied research in this area still needed to be done. It seems that confusion still exists and that practitioners are unclear how to define the value, penalty, cost or risk factors when prioritising the requirements. In addition, requirements prioritisation techniques are not scalable for a larger number of requirements; are time-consuming; are complex, therefore difficult to implement; deal mostly with small numbers of requirements and that there is no sufficient automation.

The final section presented the focus and requirements of the research.

Chapter 4

Development of the Decision Support Methodology

4.1 Introduction

As mentioned in the Introduction, product design is a complex, often ill-defined, and iterative process. It gradually becomes better defined as it progresses through the stages of product development. The design decisions made during various stages of the design process have a profound impact on the product life cycle cost. The aim of this chapter is to introduce the proposed automated design knowledge capture methodology that aids the decision making process. The methodology framework is outlined in Figure 4.1 and the data flow roadmap in Figure 4.2. The concept is based on the network interdependency model where links between the nodes serve as information carriers allowing the data to be fed in a forward or backward direction. The links between the requirements, design and cost domains capture the relational dependency information as well as the domain data which is then used for calculation and visualisation purposes. The aim is to bring the cost data forward into the requirements domain by calculating the manufacturing cost of the features that satisfy a particular requirement. The overall impact of each requirement expressed in relative terms is also calculated and displayed in the requirements domain. Furthermore, knowledge that resides in the part as well as design process knowledge, across the requirements, design and cost domains, are captured as a result. The framework can be broken down into seven stages, as shown in Figure 4.1:

1. **Data capture** (automated process) - the relevant information from the data sources are captured, structured and saved into a tabular data format (i.e. as CSV file type). The process is automated through the code using the API platform in each of the tools (data sources).
2. **Data upload** (manual process) - the relevant structured data captured from different data sources needs to be uploaded by a user into a BOXARR software.

3. **Requirements clustering** (automated process) - the saved requirements data, retaining the same data structure as in the requirements management tool is then imported to BOXARR software. The clustering is performed by code in BOXARR.
4. **Setting the importance factor** (automated process) - the key driving requirements are automatically identified by BOXARR through the criticality (importance) factor which is defined by modal verbs as well as functional requirements in the Component Requirements Document. The function written in BOXARR automatically assigns the relevant factor which is based on modal verbs.
5. **Mapping functional requirements to design features** (automated process) - functional requirements are automatically mapped to design features through the Design Failure Mode and Effect Analysis (DFMEA) data.
6. **Linking design features to cost models** (automated process) - the links from the design features to Vanguard cost models are automatically created by importing the relevant data into BOXARR from the cost data file.
7. **Calculating requirements impact** (automated process) - once all the relevant links are created, the impact of each requirement is calculated. This is done automatically by code in BOXARR. The impact is calculated as a relative value with weights put on requirements importance, manufacturing cost, requirements complexity and features complexity.

The following sections describe the methodology and provide an overview of each element of the framework. The framework is applied to three test cases presented in Chapters 5 and 6.

4.2 Requirements Document

Requirements elicitation, definition, and analysis play a critical role in the product development process. Requirements build a strong foundation for the project by forming the basis for the architectural design, integration, and verification tasks (Walden et al., 2015). Therefore, it is essential to establish good base requirements early in the project life cycle. It is important that anyone involved in the requirements elicitation, definition, and analysis would ensure that all necessary changes of the requirements are done in the early stages of the product development process. Due to the fact that requirements carry a cost, any requirement changes during later product development stages can have a significant cost impact on the project. In some cases it even leads to the cancellation of the project. In order to avoid any or unnecessary changes of the requirements in later product development cycles, requirements have to be appropriately crafted. In other words, requirements have to have certain characteristics, or attributes, that eliminate any ambiguities from the definition of the requirement. International Council on Systems

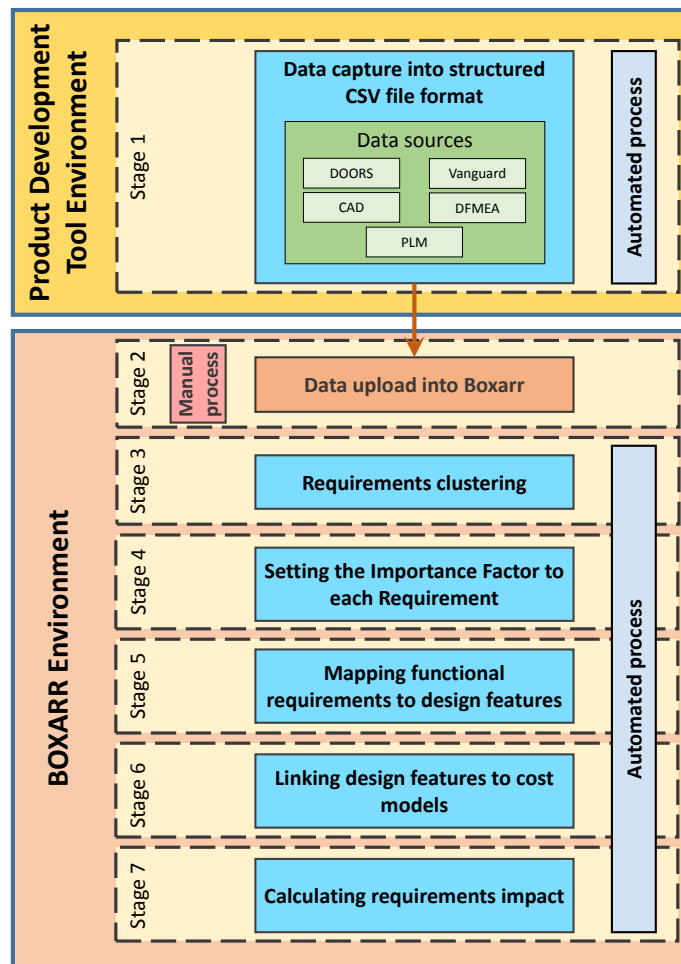


Figure 4.1: The Methodology Framework with indicated automation steps.

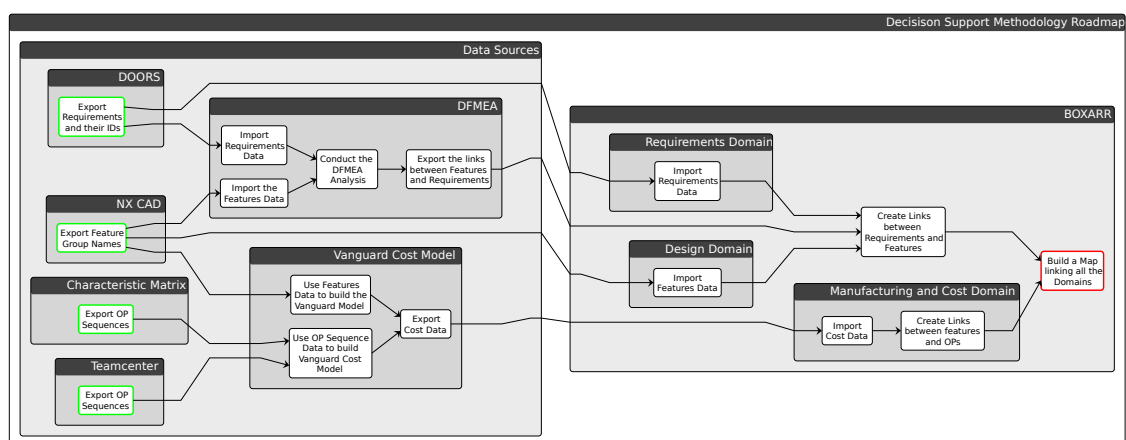


Figure 4.2: The roadmap to create links between Requirements, Design, and Cost domains

Engineering (INCOSE) proposed the following attributes that should be considered for every requirement (Walden et al., 2015):

Necessary - only necessary requirements should be written that are not redundant (i.e. covered in some other grouping of requirements) and do not prescribe the specification of design. Since every requirement generates a burden in the form of processing, maintenance, and verification, unnecessary requirements only add an unnecessary extra effort and most certainly do not add value.

Implementation Independent - the requirement has to be solution neutral and should not specify the design. In other words, the requirement should specify “what” is to be done, rather than “how” it is to be done.

Clear and Concise - requirements must be written with extreme care using a clear and exact language and should address one and only one concept. Since the key purpose of the requirement is to communicate, the language used must also be in sufficient detail in order to meet all reasonable interpretations. Any conflicting and contradicting requirements must be avoided and a glossary should be used to define the terms that could have multiple interpretations.

Complete - the specified requirement should be complete, verifiable with no need for further elaboration.

Consistent - requirements should comply with the government, industry, and product standards, specifications, and interfaces. In addition, requirements should not be contradictory or duplicated.

Achievable - the experts such as designers and manufacturing engineers as well as customers/users should participate in requirements definition to ensure the achievability of the requirements.

Traceable - requirements should be traceable to the higher level specification and/or user need.

Verifiable - when a system hierarchy is designed, each specified requirement should have a corresponding method of verification. The requirement is verified by only one of the four standard verification methods: inspection, analysis, demonstration, or test. If multiple methods of verification are required, then the requirement should be decomposed into multiple requirements. However, there is a potential to merge requirements if one method verifies multiple requirements.

To avoid misinterpretation and ambiguity of the requirements, the exact meaning of the wording should be established when defining requirements. A good example of this is the use of modal verbs. The modal verbs that are used in the Rolls-Royce’s Component Requirements Document (CRD) have the following meaning (Carter and Saunders, 2015: p. 5):

- ‘The word **SHALL** in the text denotes a mandatory requirement of this document. Departure from such a requirement is not permissible without formal agreement.’
- ‘The word **SHOULD** in the text denotes a recommendation or advice and is expected to be followed unless good reasons are stated for not doing so.’
- ‘The word **MUST** in the text is used for legislative or regulatory requirements (e.g. Health and Safety) and shall be complied with.’
- ‘The word **WILL** in the text denotes a provision, service or an intention in connection with a requirement of this document.’
- ‘The word **MAY** in the text denotes a permissible practice or action. It does not express a requirement of this document.’

The modal verbs in this research project are used to define the importance of the requirement. Each requirement carries the weight of importance according to the modal verb. The following weights are added to each modal verb that signify the criticality of the requirement (Table 4.1):

Modal Verb	Weight	Importance
MUST	9	Critical
SHALL	9	Critical
SHOULD	6	Required
WILL	3	Optional
MAY	1	Suggestion

Table 4.1: The importance of weights on modal verbs.

The added weight to the modal verb will have a positive effect on the requirements definition process. It will force the person who defines and manages the requirements to put more thought into the requirement definition process. By adding the appropriate modal verbs to the requirements, the importance will be implicitly assigned. In addition, it will provide a structured approach to the requirements definition process. Categorising requirements by importance eliminates the ambiguity factor leading to better defined requirements. Moreover, it will motivate to define only those requirements that fit into one of the five categories and will avoid unnecessary requirements.

The focus of this research is on the functional requirements since these are often the major design drivers of the product. Furthermore, functional requirements can be influenced internally within the company as opposed to regulatory requirements. In other words, the organisation has the power to change the requirements if needed, but it might not easily change the regulatory requirements. The proposed decision support framework is focused on the interrelations of requirements that have traceable links to cost. The argument has been put forward that having an automated tool that links the relevant requirements to cost and feeds the data to the requirements domain, will improve the decision making process. The main goal of the framework is to calculate the impact

of the requirements as well as the design features cost and provide the relational map between the requirements, design, manufacturing and cost domains which is generated in an automated way. Knowing the impact of each requirement as well as having visually presented links to design features, design engineers can make better trade off studies by simply changing the design feature, or manufacturing operation, or possibly merging or relaxing requirements. Therefore, the tool can be used in trade off studies as well as for knowledge capture activities. Another benefit for using the proposed framework is that it will provide the factual manufacturing process and cost data for better and more accurate decision making. It will eliminate the assumption aspect, because the cost data will be provided from the accurate cost data depository. Section 4.7 provides the detailed explanation of the methodology.

4.3 Clustering of the Requirements in BOXARR

This section introduces the BOXARR software, provides a background information on the concepts that the software is built upon as well as describing its usage as a decision support tool in product development life cycle.

4.3.1 BOXARR Software

BOXARR is a commercial software that provides the capability to solve the challenges of complexity across data intelligence aggregation, systems design and engineering, supply-chain, process management, mission/program planning and execution, and joint operations methodology (BOXARR, 2016). Its core concept is based on the yFiles paradigm (yWorks, 2016) with enhanced computational functionality for system modelling, graph analysis and simulation. yFiles is a Java-based library for visualization of data structures with diverse graph layout and labelling algorithms (Wiese et al., 2002; yWorks, 2016). BOXARR's simple modelling concept of 'Boxes' and 'Arrows' together with computational functionality for system modelling provides a powerful graph modelling capability. 'Boxes' in BOXARR represent the objects and the 'Arrows' are used to indicate relationships between these objects. 'Boxes' can be placed into multiple hierarchical 'Groups' and further grouped in multiple 'Contexts'. An example of the 'Boxes' and 'Arrows' concept is illustrated in Figure 4.3. It should be noted, however, that the purpose of this research is not about the optimisation of the graph algorithms, but to demonstrate how this concept can complement the proposed decision support framework, which is discussed in more detail in Section 4.7.

The BOXARR application was chosen as the platform for the proposed decision support framework mainly due to the following reasons:

- BOXARR is already used by Rolls-Royce plc and is a part of their standard toolset.

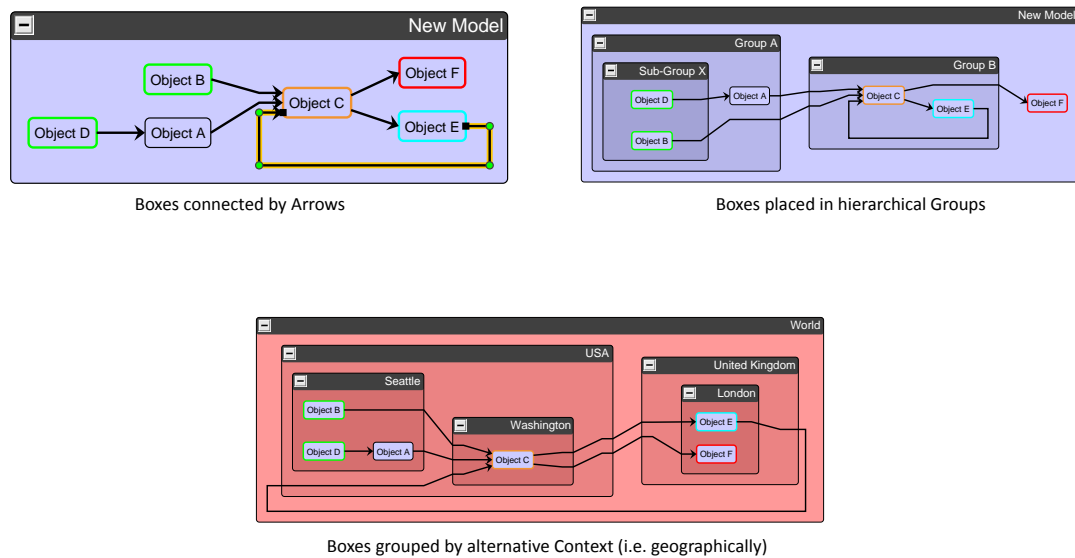


Figure 4.3: The concept of BOXARR. Adapted from (BOXARR, 2016).

- It is scalable and can be applied to the data structures of any size and complexity.
- The ability to readily identify and visualise the inherent and hidden dependencies.
- It has powerful inbuilt search and filtering tools for easy navigation and data mining.
- The platform facilitates functional analyses across data-sets associated with model objects.
- The platform can capture and store knowledge.
- The platform supports the collaborative modelling, enabling users to easily share and access information across the organisation.
- It can import data from any tool that exports tabular data.

In summary, the BOXARR application in this research is used as a tool to model and visualise the links between the product requirements, design features and manufacturing operation sequences. It is also used to calculate the impact of the requirements and the features' cost.

4.3.2 Clustering of the Requirements

Companies use different tools and software to capture, trace, analyse, and manage requirements. As previously mentioned, the aim of this research project is to validate the proposed decision support framework by using tools already in use by industry. A

generic decision support framework should permit the manipulation, and analysis of existing data for the benefit of improved decision making. The proposed novel decision support framework leverages on existing tools to improve the decision making process by bringing in the factual data element and eliminating the need for assumptions.

Most requirements management tools can easily export data into tabular data format (i.e. CSV file type). The idea is to import the requirements data (from the CSV data file) into BOXARR with the same data structure format as it was in the requirements management software. IBM® Rational® DOORS® is a primary requirements management and analysis tool at Rolls-Royce. Figure 4.4 illustrates an example of the CRD and its structure. Figure 4.5 shows an example of listed regulatory framework requirements.

As seen in the Figure 4.5, CRD has the following attributes: heading, requirement ID number, requirement description, rationale and change. The idea is to reproduce the same requirements data structure in BOXARR, with the same attributes as shown in Figure 4.4. This is achieved by importing the same requirements data file into BOXARR, which was exported from DOORS in the tabular data format. Another benefit, that BOXARR software provides, is the ability to import only those data fields that are of interest to the user. Therefore, it is possible to achieve the same CRD structure format in the BOXARR application. An example of how imported data looks in BOXARR is shown in Figures 4.6, 4.7 and 4.8. As illustrated in the figures, the requirements are clustered according to their structure in DOORS. The benefit of having the same structure in BOXARR is twofold. Firstly, there is no need to adapt to a new format of the data structure which eliminates the need 'to get used to' and increases efficiency. Secondly, the data is clustered. This, in turn, allows easy search and navigation.

4.4 Key Driving Requirements

In the literature, it is commonly stated that approximately 80% of the total requirements are perceived as critical or driving requirements (Wiegers, 2003; Berander, 2007) and about 20% of the driving requirements are responsible for nearly 80% of relational interdependencies between the requirements (Carlshamre et al., 2001). Therefore it is important to identify the key driving requirements because they define the inter-relational complexity between them. This, in turn, allows better estimation of product development costs early in the product development life cycle as complexity is often inversely correlated with cost. In other words, when more complexity is added to the product or processes, the costs increase. Additionally, if the complexity is identified later in the product development life cycle, it may lead to a substantial increase in the development costs. In this research, the criticality of the requirements is defined by the modal verbs. Therefore, the requirements that have MUST and SHALL modal verbs are identified as critical or driving requirements. Here it means, that these requirements will

Security classification Private - Rolls-Royce Data			
TABLE OF CONTENTS			
1	DOCUMENT REVISION HISTORY	4	
2	INTRODUCTION	5	
2.1	DOCUMENT SCOPE	5	
2.2	SYSTEM PURPOSE	5	
2.3	DEFINITIONS & TERMINOLOGY	5	
2.3.1	<i>Requirements Terminology</i>	<i>5</i>	
2.3.2	<i>Systems Engineering Terminology</i>	<i>6</i>	
2.3.3	<i>Using This Document</i>	<i>6</i>	
2.3.4	<i>Acronyms</i>	<i>8</i>	
2.4	SYSTEM CONTEXT	8	
3	OPERATIONAL REQUIREMENTS	9	
4	NON-FUNCTIONAL SYSTEM REQUIREMENTS	10	
4.1	COMPONENT SYSTEM REQUIREMENT	10	
4.1.1	<i>Cavity Diagram</i>	<i>10</i>	
4.1.2	<i>Cavity Connectivity</i>	<i>11</i>	
4.2	UNIT COST	12	
4.3	LIFECYCLE COST (LCC)	14	
4.4	WEIGHT	15	
4.5	LIFE	16	
4.6	SAFETY & RELIABILITY	17	
4.6.1	<i>Engine Failure Cases</i>	<i>18</i>	
4.6.2	<i>Reliability Rates</i>	<i>18</i>	
4.7	REGULATORY FRAMEWORK	19	
4.7.1	<i>CS-E VIBRATION</i>	<i>19</i>	
4.8	ASSEMBLY	20	
4.8.1	<i>Assembly & Disassembly</i>	<i>20</i>	
4.8.2	<i>Axial Setting</i>	<i>20</i>	
4.8.3	<i>Balancing</i>	<i>20</i>	
4.9	DESIGN TRADES	20	
5	KEY CHANGE PACKAGES	23	
5.1	CHANGE PACKAGE REQUIREMENTS	23	
5.1.1	<i>Using this section</i>	<i>23</i>	
5.1.2	<i>Master Programme Milestones</i>	<i>23</i>	
5.1.3	<i>Batch Points</i>	<i>23</i>	
5.1.4	<i>Deliverables</i>	<i>24</i>	
5.2	BASELINE SOLUTION(S)	25	
5.2.1	<i>Production Baseline</i>	<i>25</i>	
5.2.2	<i>Solution Milestones</i>	<i>26</i>	
5.2.3	<i>Non-Functional System Requirements</i>	<i>26</i>	
5.2.4	<i>Non-Functional Implementation Requirements</i>	<i>26</i>	
5.2.5	<i>Non-Functional Performance Requirements</i>	<i>26</i>	
5.3	DEVELOPMENT PROGRAMME SUPPORT	26	
5.3.1	<i>Development Baseline</i>	<i>26</i>	
5.3.2	<i>Development Thermal Paint Standard</i>	<i>28</i>	
5.3.3	<i>Development Strain Gauge Standard</i>	<i>29</i>	
5.4	OPTIMISATION SOLUTION(S)	30	
5.4.1	<i>MK 1.1 HPT Blade</i>	<i>30</i>	
5.4.2	<i>Non-rubbing standard of HP Turbine blade</i>	<i>32</i>	
5.4.3	<i>Reduced Lockplate loading onto coverplate</i>	<i>33</i>	
5.4.4	<i>Shank re-pressurisation</i>	<i>34</i>	
5.4.5	<i>SFC improvements for EIS</i>	<i>35</i>	

Document number EDNS010002230 80	Issue 002	Security classification Private - Rolls-Royce Data	Page 2 of 62
--	--------------	---	--------------

©2016 Rolls-Royce plc
 The information in this document is the property of Rolls-Royce plc and may not be copied, or communicated to a third party, or used, for any purpose other than that for which it is supplied without the express written consent of Rolls-Royce plc.

Figure 4.4: Example of Component Requirements Document and its structure.
 Adapted from (Carter and Saunders, 2015).

have the highest importance weight. This weight will be used to calculate the impact of each requirement. Table 4.1 in Section 4.2 summarises the relations between the modal

Security classification
 Private - Rolls-Royce Data

XWB-1000HPTB:-102 Achievement of these life targets shall demonstrate robustness to environmental and input variables that do not result in a reduction in life below the specified target. {TRACE: None}

RATIONALE:

CHANGED?:

4.7 Regulatory Framework

XWB-1000HPTB:-104 The Turbine Sub-System shall be designed to satisfy the applicable EASA CS-E requirements that are applicable at the time of application and any special conditions and issue papers likely to be incorporated by that time. {TRACE: None}

RATIONALE: Meeting certification requirements is required for market access

CHANGED?:

XWB-1000HPTB:-105 The Turbine Sub-System shall be designed to satisfy the applicable FAR33 requirements that are applicable at the time of application and any special conditions and issue papers likely to be incorporated by that time. {TRACE: None}

RATIONALE: Meeting certification requirements is required for market access

CHANGED?:

XWB-1000HPTB:-106 The Turbine Sub-System shall be designed in order to allow Aircraft certification under the applicable EASA CS-25 regulations. {TRACE: None}

RATIONALE: Airframe certification requirements can affect TURB System design (e.g. TURB Case Fire Resistance).

CHANGED?:

XWB-1000HPTB:-107 The Turbine Sub-System shall be designed in order to allow Aircraft certification under the applicable FAR25 regulations. {TRACE: None}

RATIONALE: Airframe certification requirements can affect TURB System design (e.g. TURB Case Fire Resistance).

CHANGED?:

XWB-1000HPTB:-108 The Turbine Sub-System shall achieve 420 minute ETOPS approval (405 minutes plus 15 minutes Hold), as defined by EASA and the FAA at EIS. {TRACE: None}

RATIONALE: Project requirement - engine is intended for use in a twin-jet application and maximise competitive position

CHANGED?:

4.7.1 CS-E VIBRATION

XWB-1000HPTB:-110 {TRACE: None}

RATIONALE:

CHANGED?:

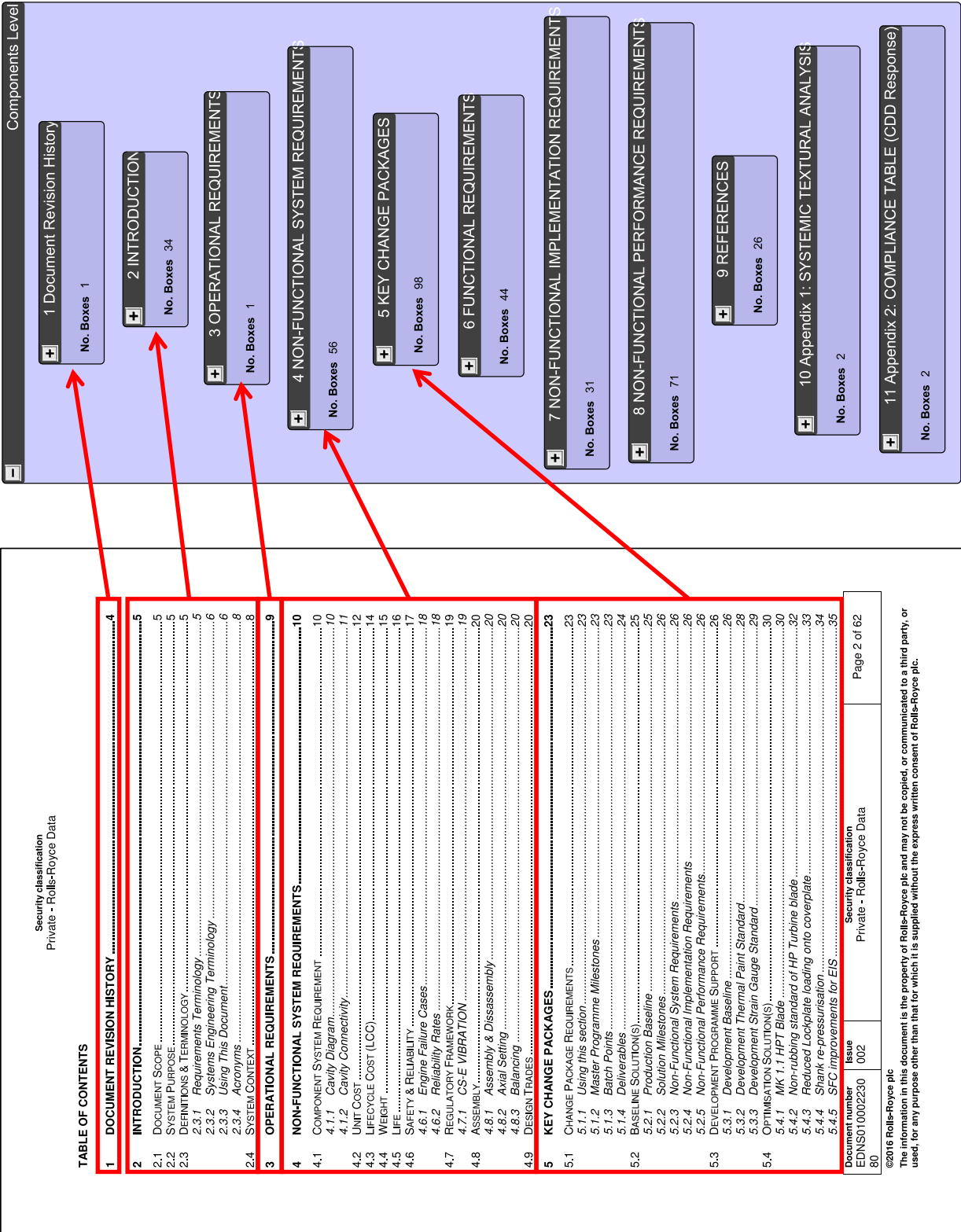
Document number EDNS010002230 80	Issue 002	Security classification Private - Rolls-Royce Data	Page 19 of 62
---	---------------------	--	---------------

©2016 Rolls-Royce plc

The information in this document is the property of Rolls-Royce plc and may not be copied, or communicated to a third party, or used, for any purpose other than that for which it is supplied without the express written consent of Rolls-Royce plc.

Figure 4.5: Example of listed regulatory framework requirements. Adapted from (Carter and Saunders, 2015).

verbs, weight and importance factors.



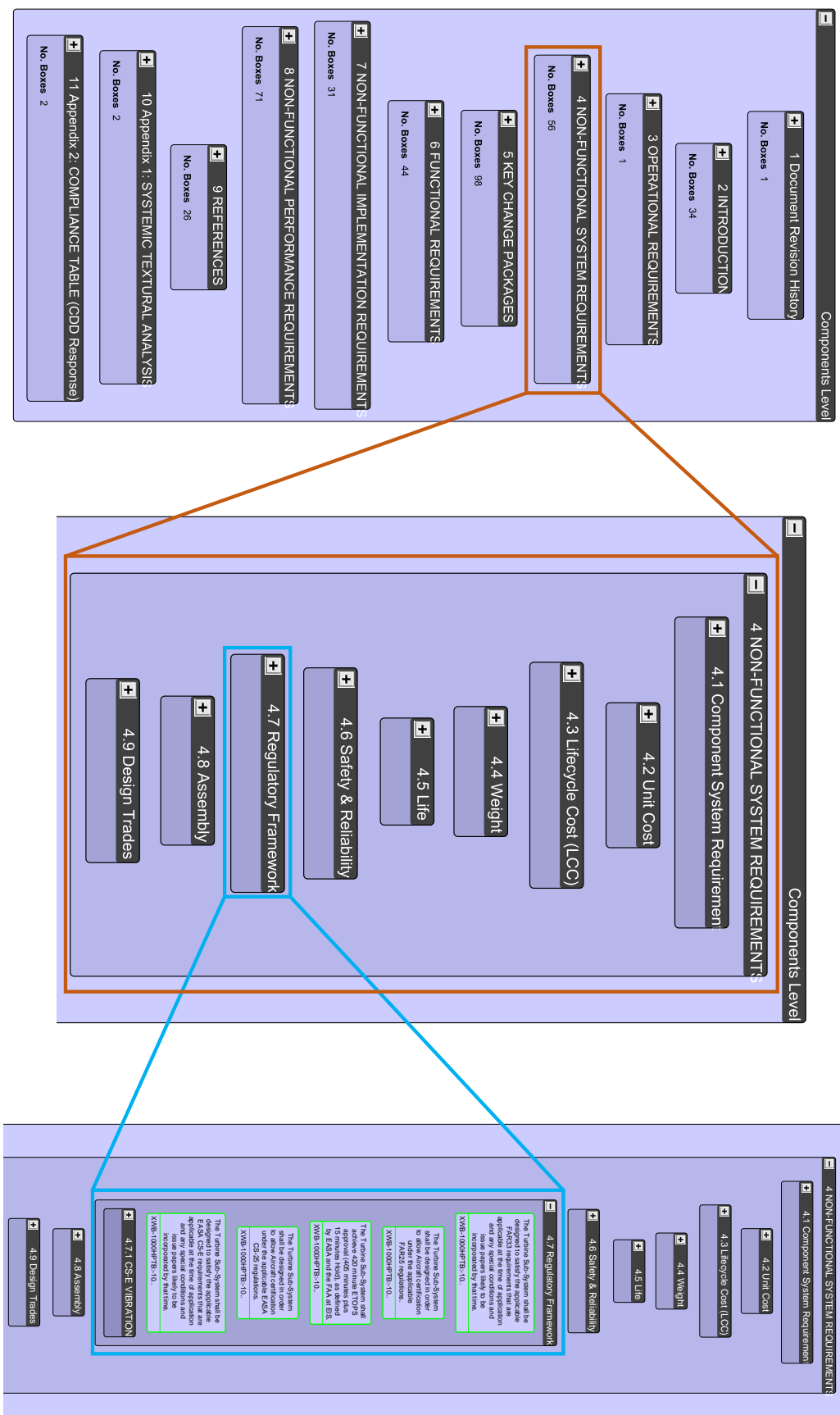
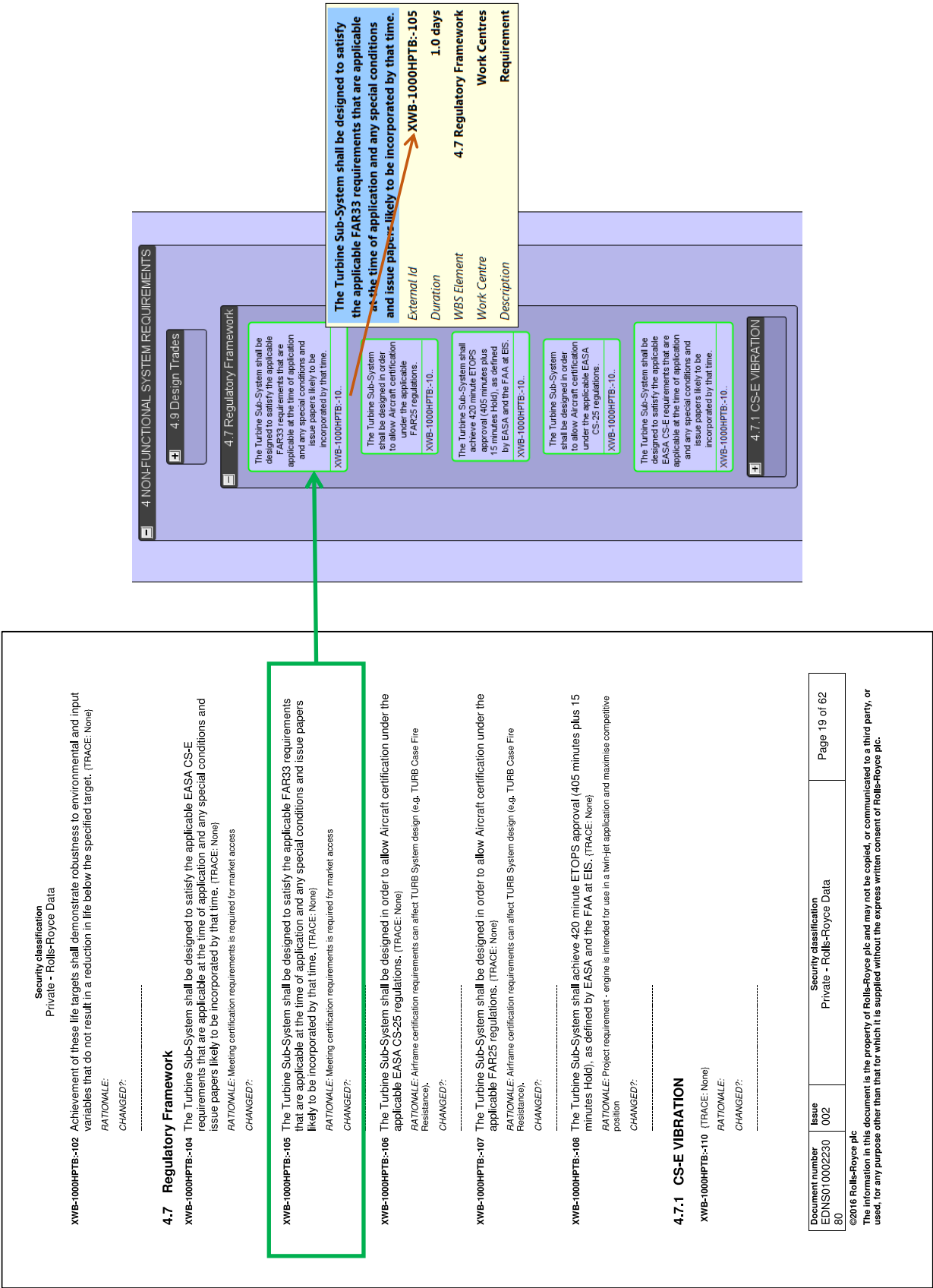


Figure 4.7: Clustered requirements in BOXARR.



4.5 Design Features

There are seven major stages in product development life cycle, namely, requirements specification, conceptual design, preliminary design, detailed design, development, production and recycle stage. Designers are involved in the majority of the product development life cycle stages. However, their main focus is towards the first four product development stages. During the conceptual design stage, the designers' task is to translate given design requirements into functions. In the preliminary design stage, functions are decomposed into corresponding geometric structures, which, in turn, are transformed into real geometries during detailed design stage (Chen et al., 2004). Geometric structures themselves can be decomposed into different features. Many different geometric solutions can be generated in order to satisfy the corresponding functions. In addition, different features can comprise these geometric structures. Design features can be interpreted as building blocks of the product. According to Shah and Mantyla (1995: p. 97) "*feature represents the engineering meaning or significance of the geometry of a part or assembly*". In this research, features are treated as building blocks of geometry and carriers of the product information. Furthermore, the geometric information of the features is used to generate manufacturing instructions in computer-aided process planning (JungHyun et al., 2000). Knowing the manufacturing processes and operation sequences, manufacturing costs can be determined. Therefore, it is safe to say that design features can be directly linked to manufacturing costs. Moreover, design features can act as the intermediary links between component requirements and cost. In this research, design features capacitate the establishment of the traceable links between the requirements domain, design domain and cost domain. Furthermore, by breaking the component into features the granularity level can be established. This can be useful in design trade-off studies. For example, by knowing the feature granularity level, a designer can modify it by merging the features or further decomposing them in order to achieve the best solution for the given constraints. By changing the feature granularity level, the direct effect on cost can be seen, since features in this research have direct links to the manufacturing cost. Similar trade-off studies can be done in the cost domain. Changing the manufacturing processes, the effect on cost can also be observed. In the author's opinion, having a relational map of requirements and design features on one screen together with the cost data improves the decision making process. As all three domains are linked, it is much easier to see how the change in one of the domains affects the others and most importantly, the unit cost of the product.

4.6 Vanguard Cost Models

Vanguard Studio is part of Rolls-Royce's standard toolset and is used to build and analyse cost models. The Vanguard System is a platform designed for decision-support

analysis and business modelling. Its *Divide and Conquer* (Vanguard, 2016) concept provides solutions to complex problems by decomposing the components of these problems into hierarchical tree structures and functional blocks. Figure 4.9 shows an example of the hierarchical tree structure. In this work, the predefined cost modelling template is used and is illustrated in Figure 4.10.

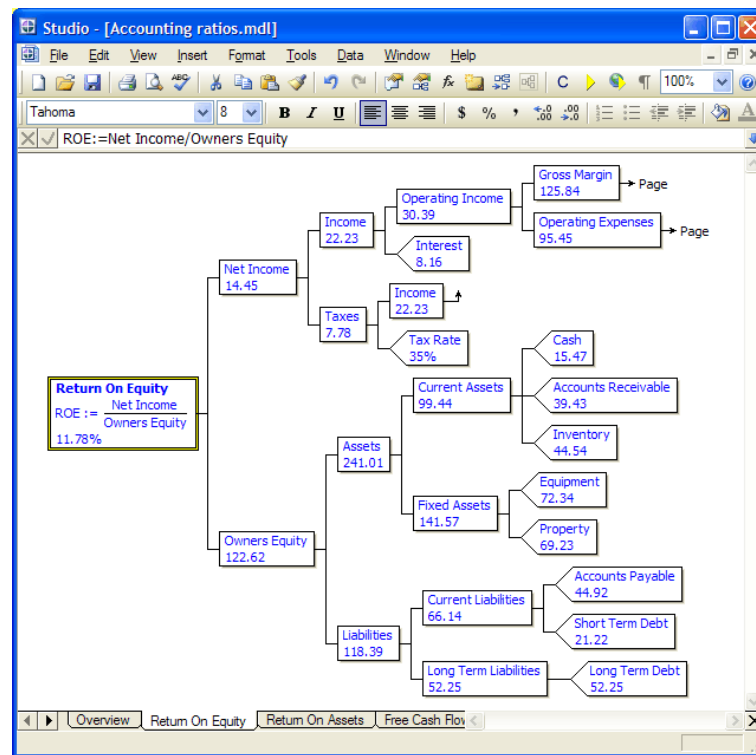


Figure 4.9: Hierarchical tree structures and functional blocks in Vanguard. Adapted from (Vanguard, 2016).

Rolls-Royce identifies eight value streams that affect the Total Unit Price. These are:

- **Raw material cost** - the cost of the raw material that is required to manufacture a part.
- **Procured parts cost** - the summed cost of the items that are required to manufacture a component. For example, the part in condition of supply (CoS) state (i.e. the start state of what will be machined by Rolls-Royce or Supplier); the finished part and the bulk material such as nuts, bolts, screws, fasteners and similar items.
- **Process cost** - the summed cost of the manufacturing operations required to convert CoS to finished part.
- **Procured service cost** - the subcontracted service cost.

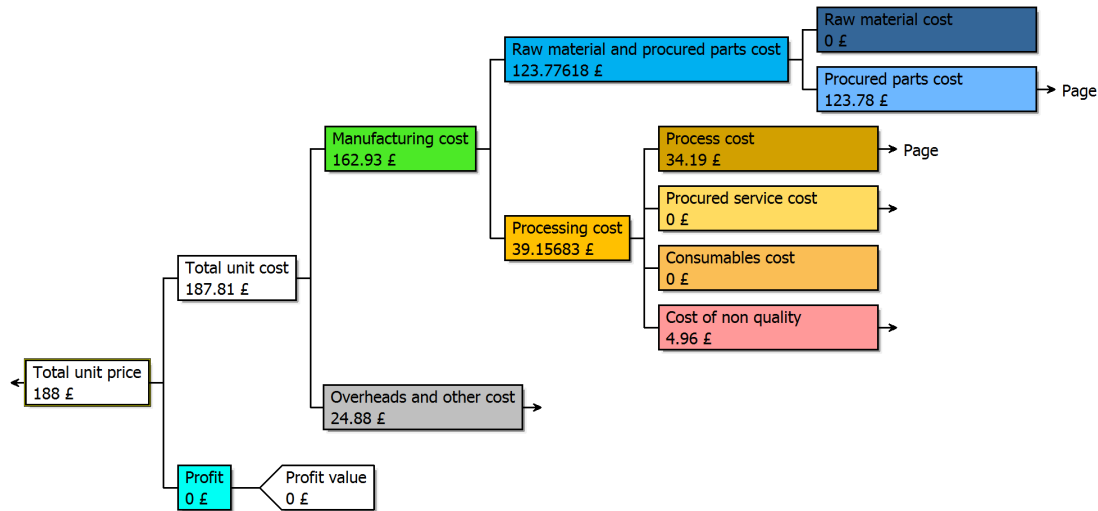


Figure 4.10: Cost modelling template used by Rolls-Royce (Rolls-Royce, 2016).

- **Consumables cost** - the cost of the items, such as tool tips, grinding wheels and coolant.
- **Cost of non quality** - scrap cost incurred by the main costs (e.g. 2% scrap incurred by a milling process).
- **Overheads and other cost** - the summed cost of packaging and logistics, purchase burden, labour burden, parts burden and overheads cost.
- **Profit** - financial gain for external suppliers, but cost to Rolls-Royce.

The visual cost structure in Figure 4.10 can be translated into mathematical expressions for each node.

$$\text{Total unit price} = \text{Total unit cost} + \text{Profit}, \quad (4.1)$$

$$\text{Total unit cost} = \text{Manufacturing cost} + \text{Overheads and other cost}, \quad (4.2)$$

$$\begin{aligned} \text{Overheads and other cost} = & \text{Packaging and logistics cost} + \text{Purchase burden cost} + \\ & + \text{Labourburden cost} + \text{Parts burden cost} + \text{Overheads cost}, \end{aligned} \quad (4.3)$$

$$\text{Manufacturing cost} = \text{Raw material and procured parts cost} + \text{Processing cost}, \quad (4.4)$$

$$\text{Raw material and procured parts cost} = \text{Raw material cost} + \text{Procured parts cost}, \quad (4.5)$$

$$\begin{aligned} \text{Processing cost} = & \text{Process cost} + \text{Procured service cost} + \text{Consumables cost} + \\ & + \text{Cost of non quality}, \end{aligned} \quad (4.6)$$

$$\text{Process cost} = \sum_i (\text{Manufacturing operation cost})_i, \quad (4.7)$$

$$\text{Procured service cost} = \sum_i (\text{Subcontracted service cost})_i, \quad (4.8)$$

$$\text{Consumables cost} = \sum_i (\text{Consumed item cost})_i, \quad (4.9)$$

$$\begin{aligned} \text{Cost of non quality} = & (\text{CoS cost} + \text{Raw material cost} + \text{Process cost} + \\ & + \text{Procured service cost} + \text{Consumables cost} + \\ & + \text{Purchase burden cost}) * \text{Scrap percent}. \end{aligned} \quad (4.10)$$

In addition to cost modelling, the software is capable of performing the following tasks ([Vanguard, 2016](#)):

- General Modelling and Problem Solving
- Collaborative Modelling

- Data Analysis
- Advanced Analytics
- Forecasting
- Decision Tree Analysis
- Sensitivity Analysis
- Monte Carlo Simulation
- Optimisation
- Application Development

The workflow of Vanguard Studio as well as inputs and outputs are shown in Figure 4.11. As seen in the figure, there are four main databases, namely design, manufacturing, purchase and resource database. During the cost modelling activities the relevant data from these databases is used as inputs to build a cost model and to calculate the unit cost. The output cost data in Vanguard is saved in tabular data format. This data is then used to create links between design features and cost in BOXARR.

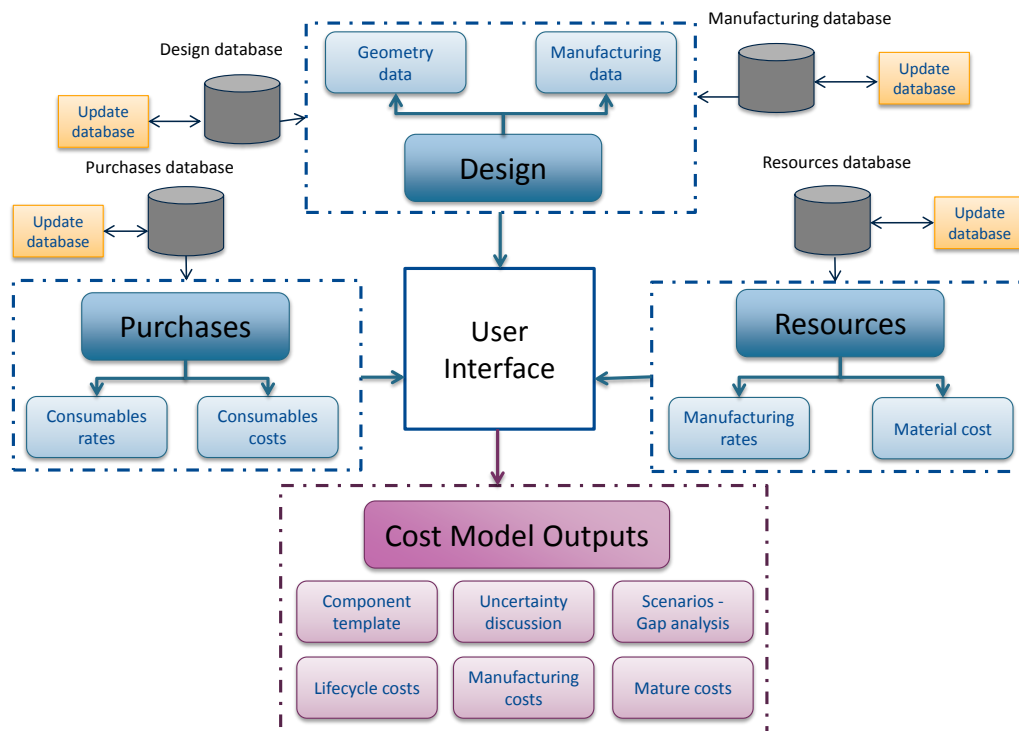


Figure 4.11: The workflow of Vanguard Studio (Rolls-Royce, 2016).

4.7 The Methodology

The aim of this section is to merge the previous sections into a coherent framework to define the methodology. The central focus of this research is the development of a method, or a tool, that would help improve the decision making process during the design and aid in capturing the knowledge of the design process. Furthermore, the main goal of this project is to bring the cost data into the requirements domain by establishing relational links between the requirements, design features and cost models, and by calculating the requirements impact. The proposed framework can be split into seven stages, as stated in the opening paragraph in Section 4.1 and illustrated in Figure 4.1. It is appropriate at this stage to recap what the stages are:

1. Automated data capture into structured CSV file format.
2. Data upload into a BOXARR software.
3. Requirements clustering.
4. Setting the importance factor to each requirement.
5. Mapping functional requirements to design features.
6. Linking design features to cost models.
7. Calculating requirements impact.

When the new product development process starts the generation of data begins. Figure 4.12 illustrates the product development life-cycle stages at Rolls-Royce. As seen from Figure 4.12 there are different activities within each of the stages. During the product development process large amounts of data are generated at each stage. However, as the project matures some of that data becomes irrelevant and is discarded or superseded by the new data. In an ideal scenario, assuming all the work has been completed to the highest standard, the data generation should be frozen at each stage. For example, at the end of the requirements specification stage, the requirements data generation should be frozen and any changes should not be necessary in the further stages of the product life-cycle. If any late changes are necessary, however, it means that requirements specification has not been done to the highest standard. As a result, the cost penalty will be incurred for late requirements change. When the data is frozen and the requirements, or design, or manufacturing process, or cost model is signed off the data is then ready to be captured. In this research, the data is captured, structured and saved in a tabular data file format for reuse by the code that is embedded in each of the data source tools. There are two approaches for data capture. First, the data can be captured and saved at the end of each stage and stored in the dedicated database. Second, the data can be captured from the source tools only when it is required. For example, when new product

development or a redesign activity starts. All the captured data has a unique identifier (Id) and is always related to the product reference number. Therefore, the traceability is established and the data can be captured and extracted from the data source at any time in the future. The second approach eliminates the data duplication and the need for extra storage.

To initiate the analysis process, the captured data for the relevant component(s) is then imported into BOXARR by selecting the appropriate data fields. Starting with requirements, every object in DOORS software has its unique ID. Additionally, requirements have the following attributes: heading, object type, description, rationale and object level. *Headings* are used to identify the requirements clusters. Component requirements are often categorised as Functional, Non-Functional and Operational requirements with various sub-categories, such as Non-Functional System requirements, Non-Functional Implementation requirements etc. The *object type* defines the type of information that is carried by the object with the unique ID. For example, in the CRD at Rolls-Royce there are six main object types: heading, requirement, information, figure, table and not set. The latter object type signifies that information type is not yet confirmed and work is in progress, while all the remaining object types are self-explanatory. A *description* attribute characterises the requirement; *rationale* attribute provides additional information about the object type and the *object level* defines the structure and hierarchy of the object types.

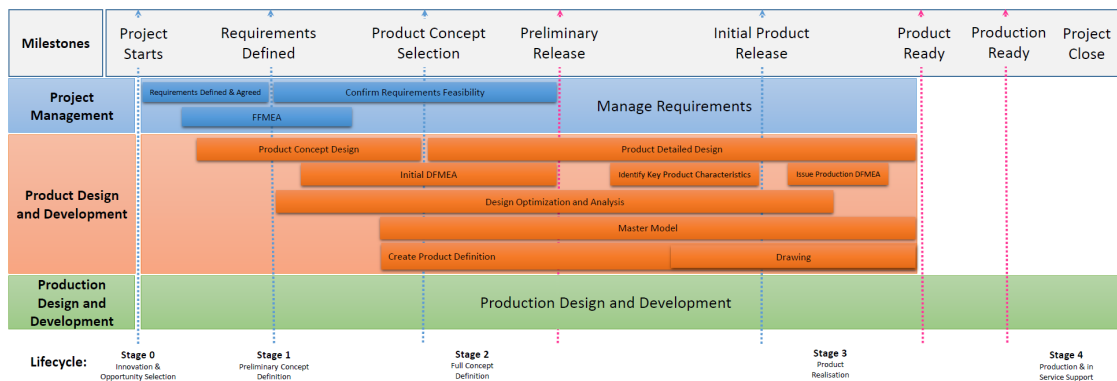


Figure 4.12: Product development life-cycle stages.

When exporting the data from DOORS, the selection option for the attributes of interest is available to choose. This proves to be a beneficial option because it eliminates the clutter of information in the CSV file. Therefore, when the file is imported into BOXARR, the relevant field name only needs to be assigned in order to match the data field in the CSV file as shown in Figure 4.13. Once the correct data is selected, BOXARR organizes the imported data into clusters (the technical term in BOXARR is called 'WBS Element') and boxes. As previously mentioned, the data structure and hierarchy is retained. Having a structured data is beneficial for the reasons mentioned

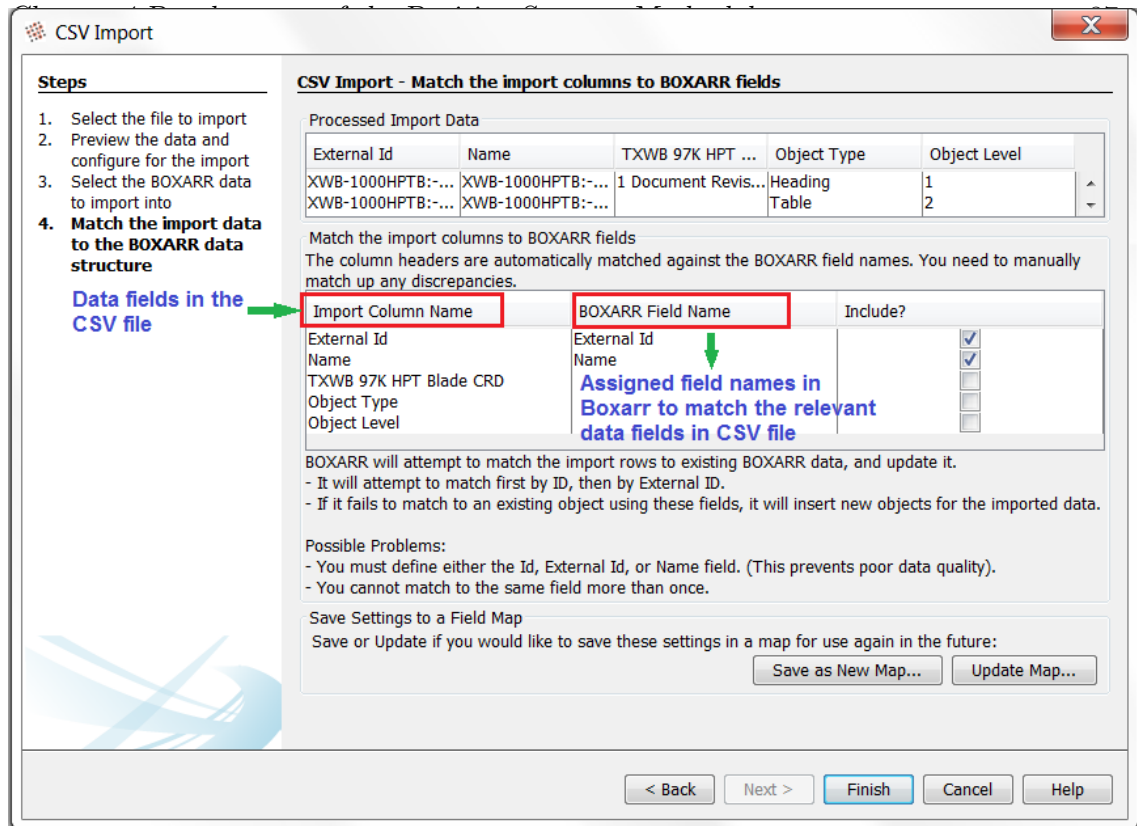


Figure 4.13: Importing data into BOXARR.

in Section 4.3.

The next step in the process is the identification of the key requirements. In a busy industrial environment, where delivery schedules are often very tight, there is always a necessity to automate the processes as much as possible. Due to this EngD research being closely related to industry, the automation aspect is considered wherever possible. However, there are a few potential issues that can lead to incorrect results when using an automated process. First, the CRD has to be well written with clear and unambiguous requirements in order to avoid misinterpretations. Second, a clear framework needs to be developed in order to define the correct importance factor for each driving requirement. Having said that, these issues, however, are not in scope of this research and are only noted to highlight the importance of well written requirements. As discussed in Section 4.2, the importance factor is defined by the modal verbs. The modal verbs used in the CRD with their corresponding weight factors are summarized in Table 4.1 but for convenience reproduced in Table 4.2.

Modal Verb	Weight	Criticality
MUST	9	Critical
SHALL	9	Critical
SHOULD	6	Required
WILL	3	Optional
MAY	1	Suggestion

Table 4.2: The importance of weights on modal verbs.

Functional Requirements, however, are often characterised without using the modal

verbs. The following solution is used in this research to overcome the problem. All the requirements that reside in the Functional Requirements section are classed as critical and thus have the highest importance factor - 9. The importance weights are assigned through BOXARR using a script that has been written to automate this procedure once the requirements data is in BOXARR. The result is then displayed in the designated data field in BOXARR GUI as illustrated in Figure 4.14.

Design features and cost data are imported from NX CAD and Vangaurd Studio, respectively, following the same procedure that was used to import the requirements data. There are two ways to save the features information in a tabular data format. First, extract them directly from the CAD data file by writing a code that does this through the CAD's API. Second, extract them from the designated features database. However, the first option to group the modelling features that define the design feature of interest is not a common practice at Roll Royce as discussed in Section 3.2. At present, the standard Rolls-Royce practice is to store Design feature information in the designated features Database. Any method that could save features into tabular format can be used. However, to avoid any feature naming errors and mistakes the feature should have a single standard name that is used globally. Therefore, in this work, the features are captured from the NX CAD by the code (Appendix B). Once the features are saved into the tabular data format the file is then ready to be imported and populated in BOXARR platform.

The cost data is stored on Vanguard Cost Modelling System (VCMS) web server. The unit cost structure in System, Sub-system and Component levels is illustrated in Figure 4.15.

A predefined component template is used for cost modelling. The component template report shown in Figure 4.16 characterises seven value streams (note that in this report Consumables Cost is included in the Process Cost). This information is used to calculate the manufacturing cost of the features as well as the impact of the requirements. In this research special attention is paid to Process Costs, because these costs, together with Material Cost, are often the biggest contributors to component unit cost as the bar chart shown in Figure 4.16 indicates. The process cost is defined by the manufacturing processes and operation sequences. A representative example of operation sequences in a tabular format is shown in Figure 4.17.

The idea is to use the cost data from the aforementioned value streams and organise it in such a way that, when imported into BOXARR, the links between the design features and relevant cost data are automatically created and the cost of each feature is calculated. Code (Appendix C) has been written that enables to organise the cost data and export it into the CSV file.

The final step in this framework is the calculation of the requirements impact. In order to do this, links between the requirements and design features should be created. The automatic link creation is achieved for Functional Requirements only, since these are the major design drivers and can be managed internally. The information and data from the

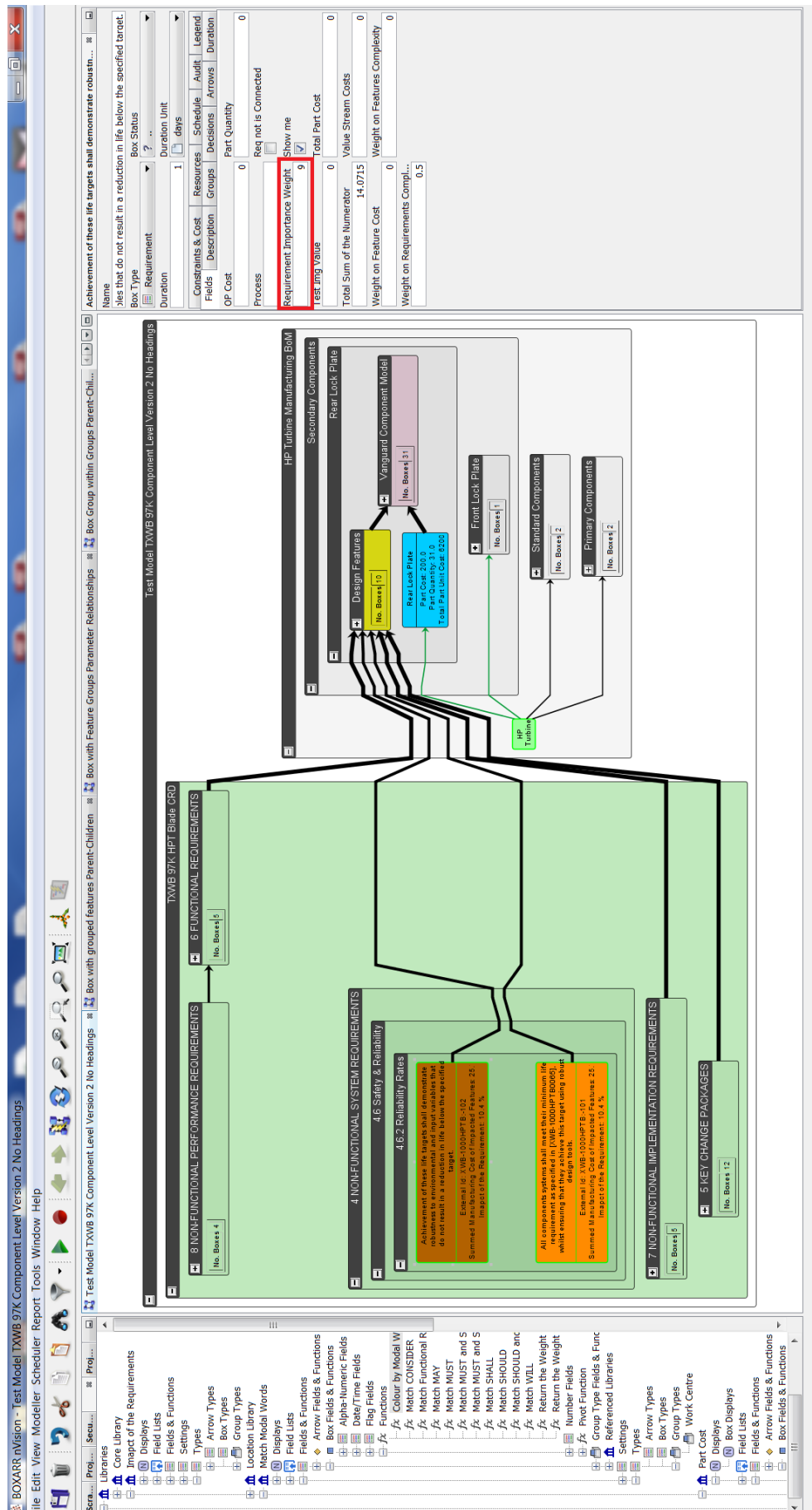


Figure 4.14: The *Requirements Importance Weight* is assigned automatically by BOXARR and displayed in the designated data field (red rectangle).

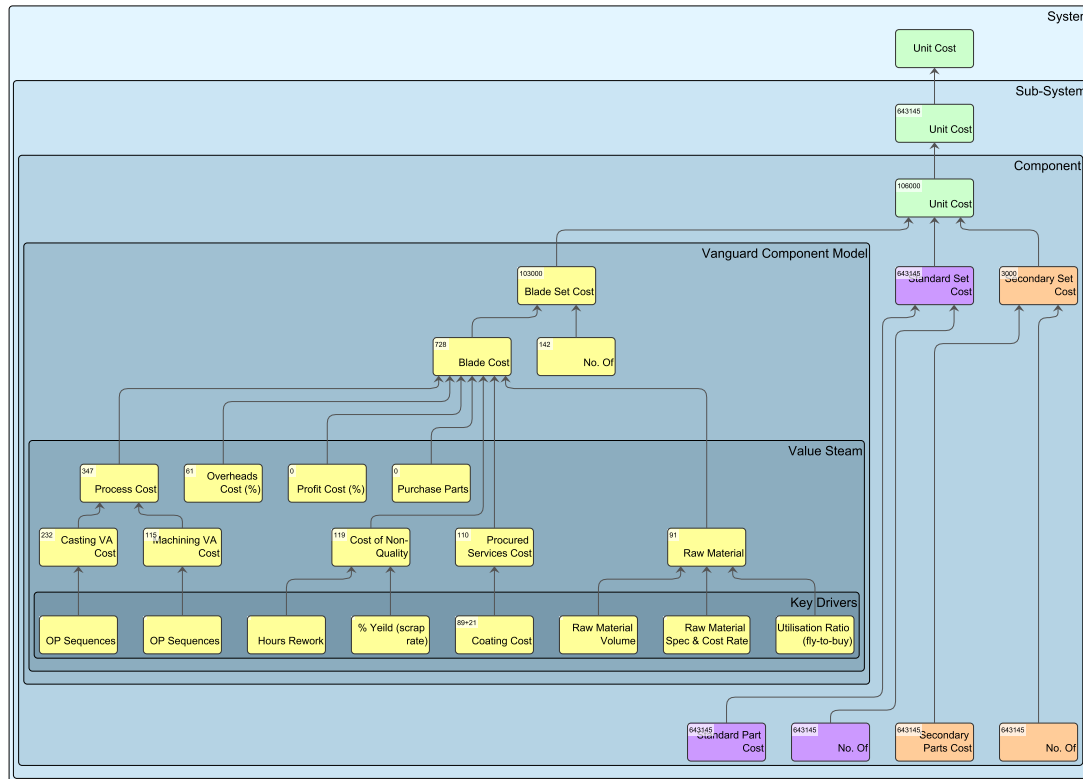


Figure 4.15: The unit cost structure in System, Sub-system and Component levels.

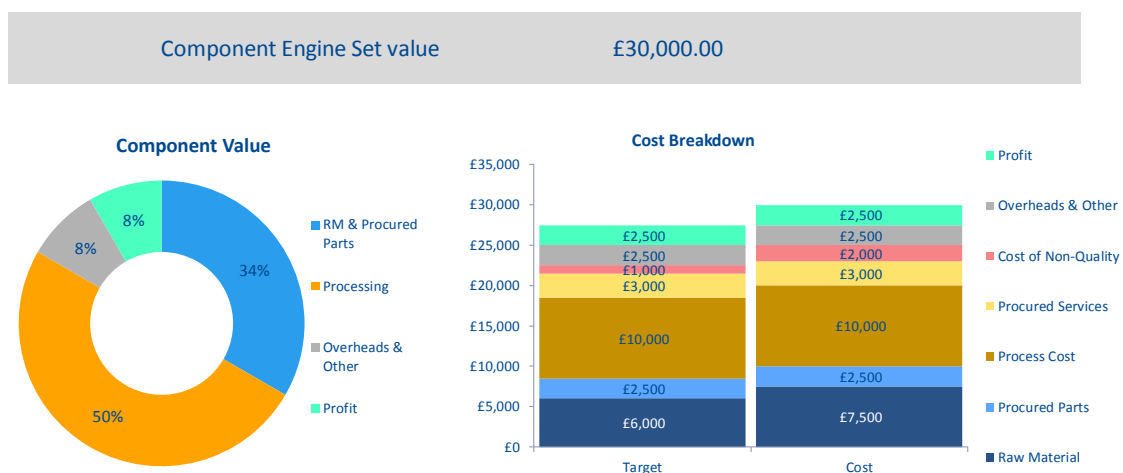


Figure 4.16: Representation of the component template report that characterizes 7 value streams (Rolls-Royce, 2016).

Op No	Op Description	Op Total Cost
Op 0200	ALIGN	£ 14.0
Op 0300	MACHINE & INSPECT	£ 212.0
Op 0500	CUT-OFF	£ 19.0
Op 0600	MACHINE & INSPECT	£ 133.0
Op 0700	UNLOAD AND INSPECT FORMS	£ 12.0
Op 0900	EDM - DIE SINK	£ 233.0
Op 1000	CLEAN AND DRY	£ 10.0
Op 1100	INSPECT ON CMM	£ 58.0
Op 1120	AQUEOUS CLEAN	£ 45.0
Op 1150	WELD HOLES	£ 48.0
Op 1200	EDM HOLES	£ 600.0
Op 1390	FILM COOL OVERCHECK	£ 6.0
Op 1400	AQUEOUS CLEAN	£ 5.0
Op 1500	WELD HOLE	£ 1.0
Op 1600	EDM GALLERY BLEED HOLE	£ 2.0
Op 1700	GRIND	£ 4.0
Op 1800	BREAK SHARP EDGES	£ 1.0
Op 1900	CLEAN AND DRY	£ 1.0
Op 1950	WELD INSPECT	£ 0.43
Op 4150	CONCESSION CHECK	£ 0.87
Op 4200	PART MARK ENGRAVE	£ 1.54
Op 4250	LOADING OF PROCESS LOT	£ 0.19
Op 4300	DESPATCH (TO PRAXAIR)	£ 0.16

Figure 4.17: Representation of the manufacturing operation sequences and their cost.

Design Failure Mode and Effect Analysis tool enables the automated link creation between the requirements and design features. As this tool is used to analyse the relations of the requirements and design features as well as functional failures, the resultant data can be used in this methodology eliminating the need to perform the same analysis twice.

To calculate the impact of each requirement, the following equation is used:

$$I_n = \frac{R_{C_n} + FC_{W_n} + F_{C_n}}{\sum_{n=1}^m (R_{C_n} + FC_{W_n} + F_{C_n})} \quad (4.11)$$

where,

I - Requirement Impact,

R_C - Requirement Links Complexity,

FC_W - Features Cost Weight,

F_C - Feature Links Complexity,

$i \in$ affected features,

n - requirement number,

m - number of key requirements.

Requirement Importance Factor (R_I) is defined by the modal verbs and the functional requirements. The latter will always carry the highest weight which is 9. The value of requirements importance factor has been chosen arbitrarily by the author. The reason why the modal verbs *MUST*, *SHALL* as well as all *Functional Requirements* have an equal weight is because these requirements have been perceived as equally important by the engineers at Rolls-Royce. However, any other value can be chosen. The importance factors for modal verbs are summarised in Table 4.1.

Feature Cost Weight (Eq. 4.12) is a product of the *Normalised Feature Cost* and *Requirement Importance Factor*. The *Normalised Feature Cost* (Eq. 4.13) is obtained from the cost data by normalising the manufacturing cost of each design feature ($Cost_i$) by the total process cost. In order to find the manufacturing cost of each feature, the relevant operation sequence costs are added (Eq. 4.7). The process cost is obtained directly from the Vanguard cost model.

$$FC_{W_n} = \sum_i (W_{C_i}) * R_{I_n} \quad (4.12)$$

$$W_{C_i} = \frac{Cost_i}{Process\ cost} \quad (4.13)$$

where,

W_{C_i} - Normalised Feature Cost,

R_I - Requirement Importance Factor,

$Cost_i$ - Cost of each feature,

$i \in$ affected features,

n - requirement number.

Requirement Links Complexity (Eq. 4.14) is the product of *Requirement Importance Factor* and *Requirement Link Weight*. The latter is defined as the number of affected design features by the requirement normalised against the total number of features that are linked to the requirements (Eq. 4.15).

$$R_{C_n} = R_{CW_n} * R_{I_n} \quad (4.14)$$

$$R_{CW_n} = \frac{|i_n|}{|j|}, \quad i \subseteq j \quad (4.15)$$

where,

R_{CW_n} - Requirement Link Weight,

$i \in$ affected features,

$j \in$ all features,

n - requirement number.

Similarly, *Feature Links Complexity* (Eq. 4.16) is the product of the *Feature Link Weight* and *Requirement Importance Factor*. The *Feature Link Weight* (Eq. 4.17) is defined by normalising the number of links from each feature to the requirements against the total number of requirements that are linked to all the features.

$$F_{C_n} = \sum_i (W_{F_i}) * R_{I_n} \quad (4.16)$$

$$W_{F_i} = \frac{|p_i|}{|p_{ij}|} \quad (4.17)$$

where,

W_{F_i} - Feature Link Weight,

p - number of affected requirements,

$i \in$ affected features,

$j \in$ all features,

n - requirement number.

As stated by the [Equation 4.11](#), each requirement is weighted against the *Requirement Importance Factor*, *Requirement Links Complexity*, *Feature Cost Weight* and *Feature Links Complexity*. To calculate the requirements impact, the weighted sum of each requirement is divided by the total weighted sum of all the key requirements. The proposed approach eliminates the speculation aspect when doing the prioritisation exercise and uses the factual data to calculate the weights as well as the impact of each requirement. Chapters [5](#) and [6](#) demonstrate the proposed methodology using the Rear Lock Plate, IPT Stub Shaft and the system of components that represent the Triple Seal Flange Joint as example test cases.

Chapter 5

Example Test Case: Rear Lock Plate

This section demonstrates the application of the proposed methodology. It is often desirable to test new methodology on a simple component and progress to more complex ones. This approach allows to test the methodology on less complex parts first and then improve the tool by applying it to more complex parts. The Rear Lock Plate has been chosen as a test case. The chosen component is not very complex from a geometry point of view, however, the interface with other components is fairly complex. The Rear Lock Plate interfaces with the following components: Seal Plate, HPT Blade, Damper, and HPT Disc. The fact that the geometry of the lock plate is simple means that not many features comprise the component. However, the relational complexity of the requirements can be observed due to the fact that the lock plate interfaces with other components. Therefore, the Rear Lock Plate is considered to be a reasonable component that will be used for testing, verification and validation of the proposed decision support tool. The following sections introduce the Rear Lock Plate and demonstrate the application of the methodology.

5.1 Rear Lock Plate

As mentioned in Section 3.2.1, the Rear Lock Plate is part of the Trent engine High Pressure Turbine sub-system (Figure 5.1). The primary and secondary function of the lock plate is to provide axial retention for the Turbine Blade and to seal the local Air System, respectively. The location of the rear lock plate is shown in Figure 5.2. The rear lock plate is composed of ten design features which are listed in Figure 5.3.

As already mentioned, the Rear Lock Plate is part of the HPT sub-system, therefore, its requirements are populated together with HPT Blade requirements. A total of 20 requirements have been identified by the designer who was responsible for the rear lock plate's design. The relations between requirements were also identified and summarised in the table shown in Figure 5.4. Number '1' signifies that the requirement is linked to a particular feature.

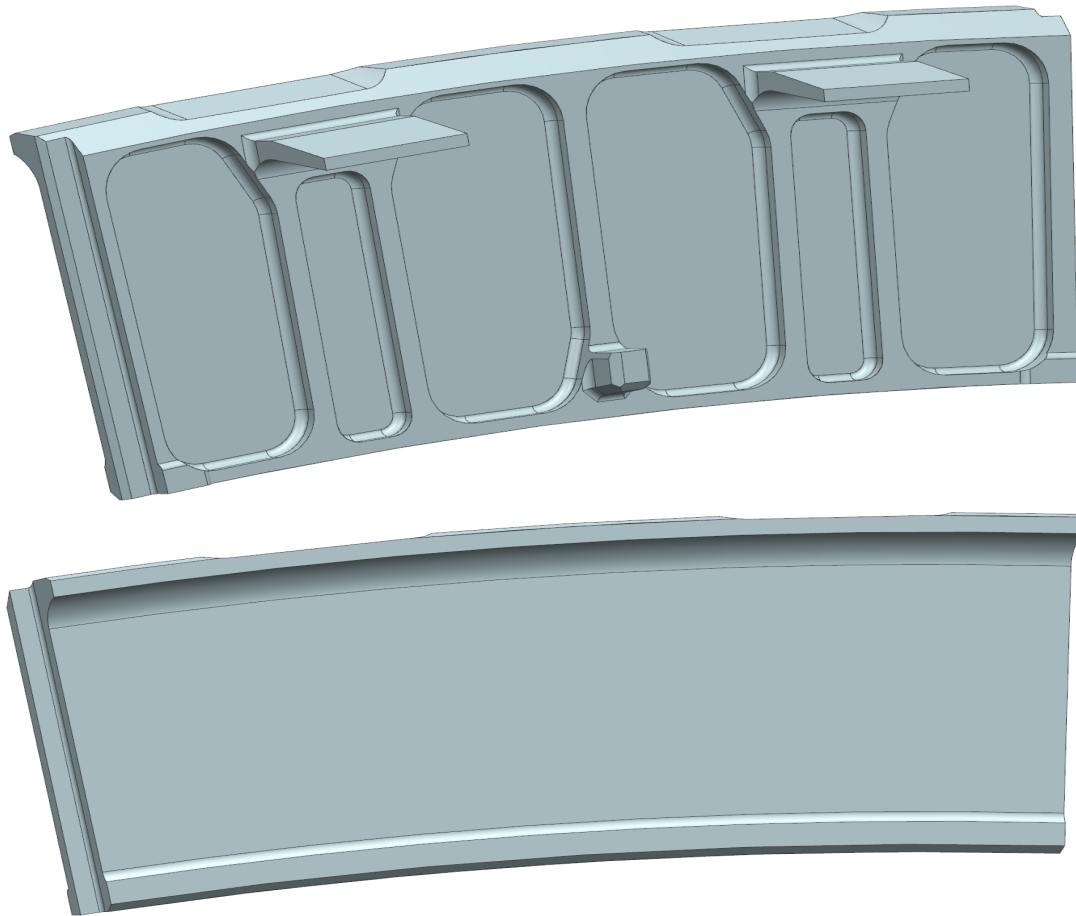


Figure 5.1: The CAD model of the Rear Lock Plate.

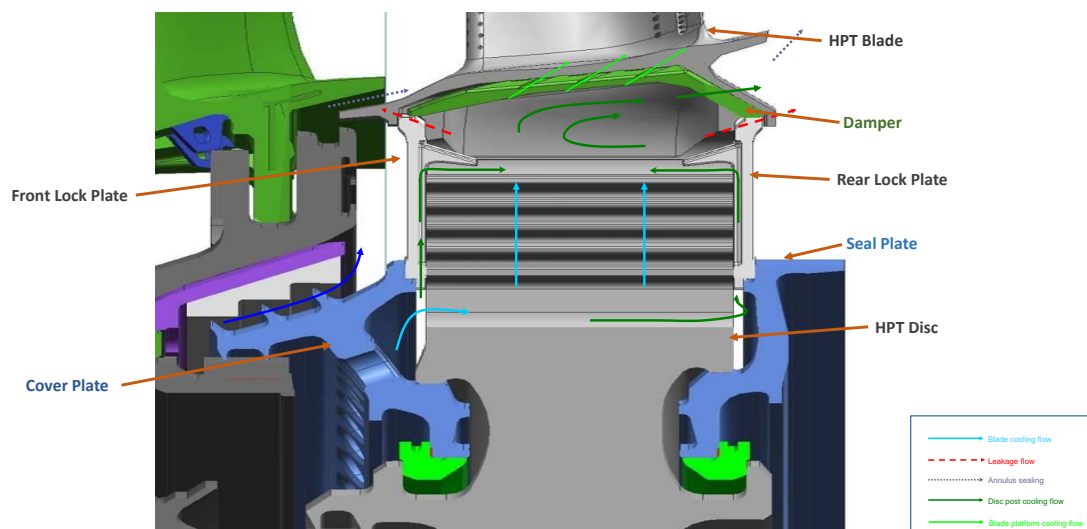


Figure 5.2: The components that rear lock plate interfaces with.

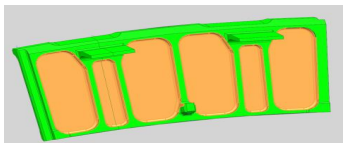
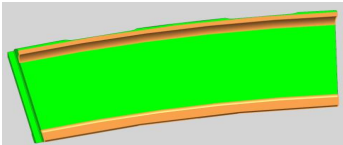
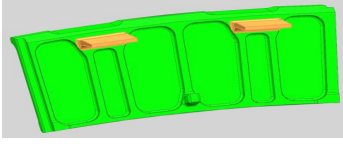
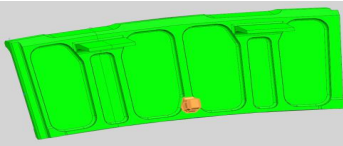
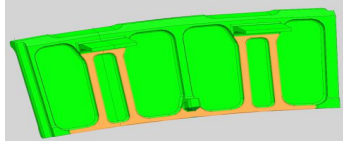
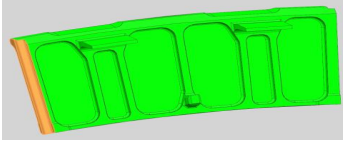
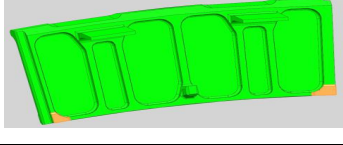
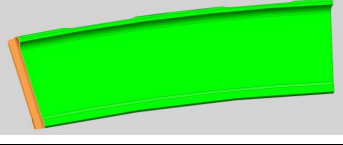
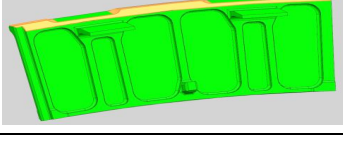
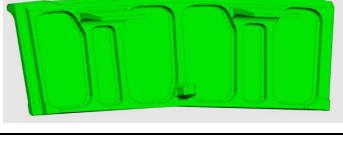
Feature	Name	Feature	Name
	Weight Saving Pockets		Sealing Faces
	Deflectors		Anti-Rotating Tang
	Stiffening Ribs		Overlap Feature (Inner)
	Foot Contact Pads		Overlap Feature (Outer)
	Pressure Pads (Top)		Material

Figure 5.3: Design features of the rear lock plate.

The Vanguard Cost Model for the Rear Lock Plate has been built and is shown in Figure 5.5. The Rear Lock plate is manufactured using the processes of casting and machining. But only a machining cost model is used for analysis in this test case example. The casting cost is included into the *Procured Parts and Services* cost field. *Process Cost* is defined by the costs of the operation (OP) sequences that are required to manufacture the features of the lock plate. The representative OP sequences are shown in Figure 5.6. The cost data for the Rear Lock Plate has been saved into the tabular data format and used to create the relational graph. The method demonstrating how to build the relational graph is described in the following section.

5.2 Methodology Demonstration

To satisfy the requirements, manufacturing and cost constraints at the same time is nearly impossible. Therefore, some compromises have to be made. In such situations the trade-off studies are always necessary in order to achieve the best compromise that results in the maximised product design. In the aerospace industry, there is a constant battle between performance and cost (Nolan et al., 2016). Customers want the jet engine to be efficient, meet the performance criteria and be reasonably priced. They anticipate that an efficient and well performing jet engine will reduce their operational costs. On

Requirements ID	Description	Rear Lock Plate Design Features									
		Weight Saving Pockets	Deflectors	Stiffening Ribs	Foot Contact Pads	Pressure Pads (Top)	Sealing Faces	Anti-Rotating Tang	Overlap Feature (Inner)	Overlap Feature (Outer)	Material
XWB-1000HPTB:-101	All components systems shall meet their minimum life requirement as specified in [XWB-1000HPTB0065], whilst ensuring that they achieve this target using robust design tools.	1	1	1		1					1
XWB-1000HPTB:-102	Achievement of these life targets shall demonstrate robustness to environmental and input variables that do not result in a reduction in life below the specified target.	1	1	1		1					1
XWB-1000HPTB:-245	1.2.5 Transfer air from...Rear lockplate cavity (5) to...Shank cavity (6)		1				1		1	1	
XWB-1000HPTB:-475	The robustness to ingestion at the damper shall not be compromised from the baseline solution and therefore pressure shall be equalised at the inboard radius of the front lockplate. The hole size shall be approved by SAS and sub-system in a DDIS.						1		1	1	
XWB-1000HPTB:-232	1.1.5 Guide air through Rear lockplate cavity (5)						1		1	1	
XWB-1000HPTB:-246	1.2.6 Transfer air from...F1 duct rear outlet cavity (10) to...Rear lockplate cavity (5)						1		1	1	
XWB-1000HPTB:-263	3.1.4 Seal leakage path from...Rear lockplate cavity (5) to...Rear damping cavity (4)						1		1	1	
XWB-1000HPTB:-499	The leakage through the shank cavity shall be less than 4.5mm ² per lockplate at the lockplate foot.						1		1	1	
XWB-1000HPTB:-304	Minimal thickness lockplates shall be considered that are sized on lockplate buckling.	1		1							1
XWB-1000HPTB:-500	The leakage through the shank cavity should be less than 1mm ² per lockplate at the lockplate foot.						1		1	1	
XWB-1000HPTB:-313	Consider tight clearance/slight interface fits of the lockplate within the Blade groove				1	1	1	1			
XWB-1000HPTB:-350	1.1.5 Guide air through Rear lockplate cavity (5)						1		1	1	
XWB-1000HPTB:-365	1.2.5 Transfer air from...Rear lockplate cavity (5) to...Shank cavity (6)		1				1		1	1	
XWB-1000HPTB:-366	1.2.6 Transfer air from...F1 duct rear outlet cavity (10) to...Rear lockplate cavity (5)						1		1	1	
XWB-1000HPTB:-378	3.1.4 Seal leakage path from...Rear lockplate cavity (5) to...Rear damping cavity (4)						1		1	1	
XWB-1000HPTB:-301	The lockplate and blade interface shall achieve significant radial engagement of lockplate and Blade and avoidance of lockplate radial chocking or loss of overlap at extremes of operation. The need for complex analysis and statistical approaches should be avoided.					1		1			
XWB-1000HPTB:-467	The coverplate lift off force contribution from the lockplate shall be zero						1				
XWB-1000HPTB:-314	Consider angled Blade grooves to slide and load the lockplate into sealing engagement. Consider the necessary friction angle to permit sliding under full CF load.					1					1
XWB-1000HPTB:-231	1.1.4 Guide air through Rear damping cavity (4)		1								
XWB-1000HPTB:-297	Any radial Radial disc post cooling flows should be provisioned by grooves within the disc post or Blade hardware, avoiding excessive lockplate thickness.			1							

Figure 5.4: The summary of the rear lock plate requirements.

the other hand, the company that manufactures the jet engine is determined to build an efficient and well performing engine at the lowest cost. However, Rolls-Royce makes money in service by selling Power to the customer. The ‘*Total Care*’ contracts account for about 90% of sales. Therefore, reduced costs and increased reliability increases profit

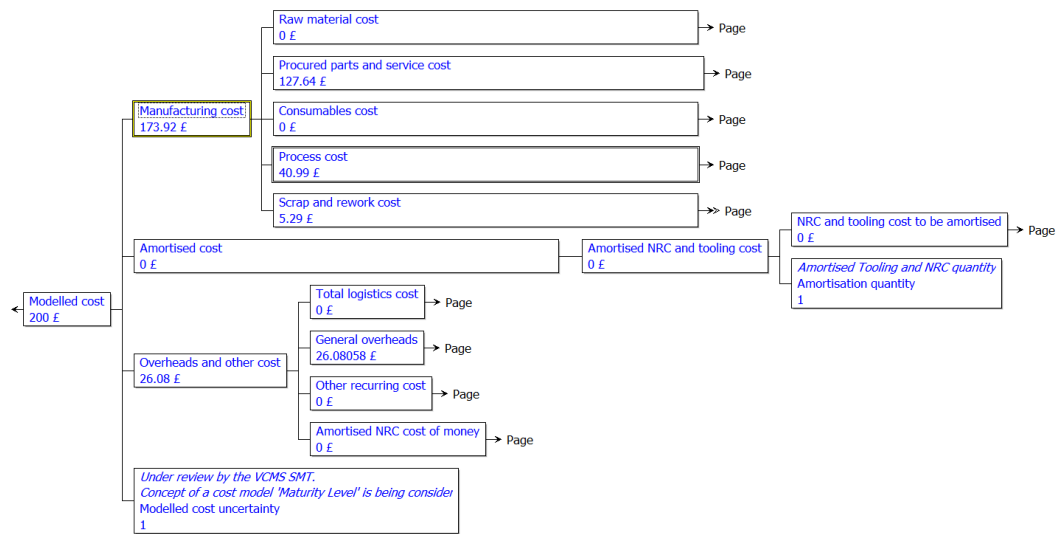


Figure 5.5: Representative Vanguard Cost Model of the Rear Lock Plate (Rolls–Royce, 2016).

margins. This, in turn, defines the success of the company.

The main focus of this research is to investigate the relations between the requirements and cost. Knowing the relational dependencies between the requirements and cost as well as the overall impact of the requirements, better decisions can be made during the trade-off studies and cost optimisation activities. The application of the decision support methodology, presented in Section 4.7, is described in the following paragraphs using the rear lock plate as a test case.

It is assumed that thorough requirements elicitation process has been done, design features of the component have been identified and the component cost model has been built. To start the analysis, requirements domain, design domain and cost domain have been created in BOXARR. As previously noted, only data in the requirements domain has been exported from the requirements management software and saved in a tabular data format. The excerpt from the CSV file is shown in Figure 5.7.

As a first iteration, the design features domain has been generated manually. The code in Appendix C extracts the cost data from the Vanguard cost models automatically. This data was then used to create the cost domain in BOXARR. The structure and hierarchy of the model was created using the data from the Vanguard cost model. The preview of the three domains and their structure is shown in Figure 5.8. As seen in the figure, Requirements are structured in the same way as they appear in the DOORS software; 10 design feature boxes were created in the Design Feature tree; and the cost data is structured according to the unit cost structure shown in Figure 4.15.

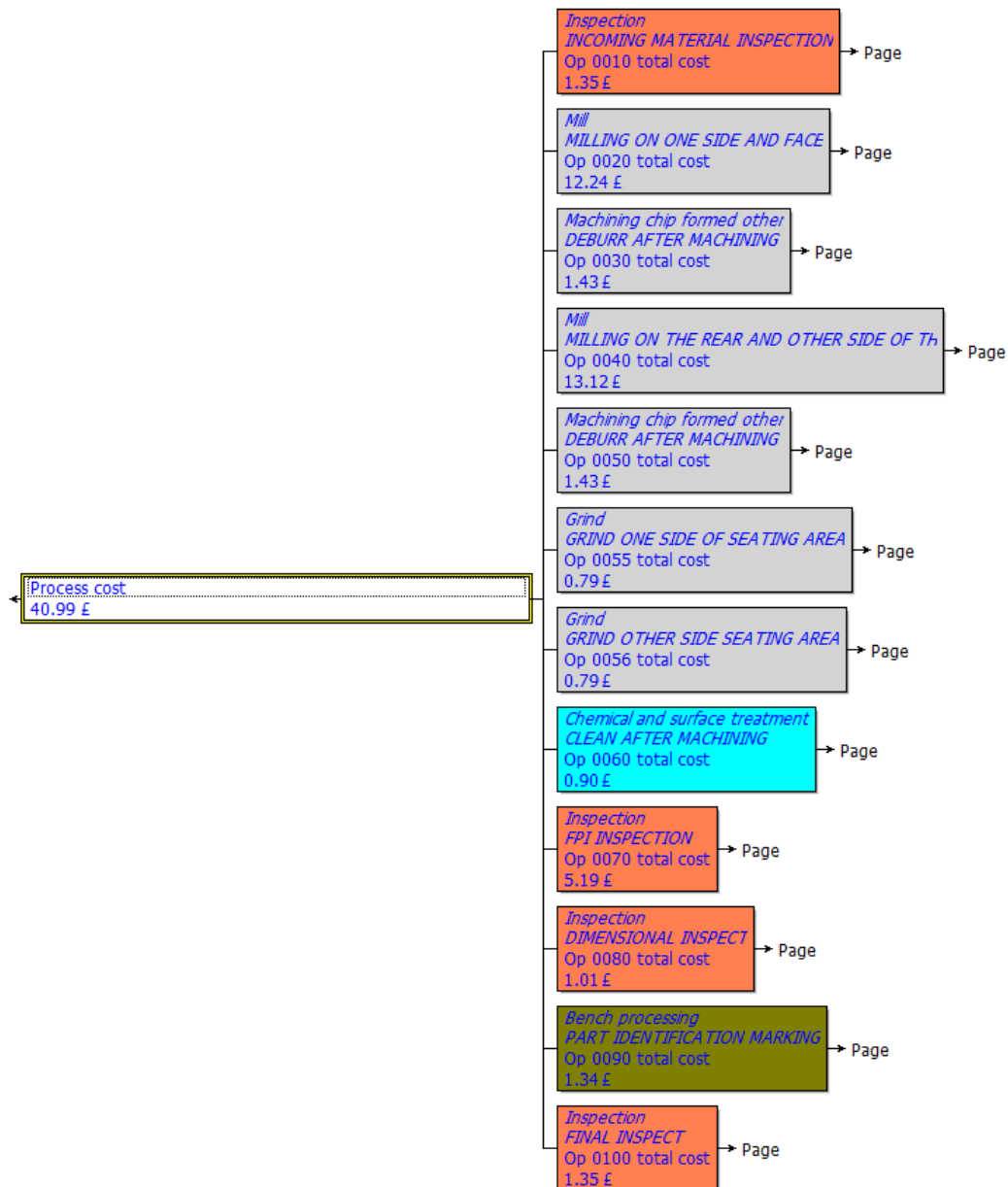


Figure 5.6: The representative operations required to manufacture the features of the lock plate (Rolls–Royce, 2016).

To maximise BOXARR’s capability and to make relational graphs more visually appealing, colours and different shapes are added to the boxes and WBS Elements. Additionally, it helps to distinguish between the three domains. In this test case, the following colour and box attributes have been applied:

- the WBS element of the Requirements domain is green and the shape of each box is a rectangle with rounded corners;
- the WBS element of the Design Features domain is dark yellow, the shape of each box is a hexagon and the boxes are coloured in yellow;

Figure 5.7: Component requirements in CSV file format.



- the WBS element of the Vanguard Component Cost Model is dark pink, the shape of each box of OP sequence is an octagon coloured in pink, and
- the shape of each box of the key drivers is a parallelogram coloured in yellow.

All of the colours and box shapes can be saved in a template for future use. This is achieved by creating the box type in BOXARR with the desired shape and colour properties. Furthermore, images can be added to the boxes. This capability is used to show the graphical representation of the design features. The comparison of the Design Features domain with and without the pictures is shown in Figure 5.9.

The ability to switch between the textual content and the graphical content saves a lot of time for a designer or engineer who is doing the analysis because there is no need to separately search for the pictures of each design feature or open the CAD tool. Once it is coded in the BOXARR software, the switch can be completed instantly by the press of the button. The arrows that characterise the relations between the two boxes can also be customised by the user. This is achieved by creating the Arrow Type with desired shape and colour properties. The following arrow types are created in this test case example: the Requirements to Features Arrows are coloured in magenta, Features to Op Sequence Arrows are coloured in yellow, and Value Stream to Final Cost Arrows are coloured in red.

The next step in the process is the identification of the requirements importance factor. Once the requirements data is imported into BOXARR, it automatically identifies the requirements importance by matching the modal words and the functional requirements. The MUST and SHALL modal verbs as well as all functional requirements carry the highest weight of 9 which is automatically assigned to each requirement and saved in the 'Requirements Importance Weight' data field in the BOXARR GUI. The requirements importance weights will be used to calculate the requirements impact. Since the Rear Lock Plate is a part of HPT sub-system, its requirements are populated together with HPT Blade requirements. As a first iteration, the requirements relevant to the rear lock plate have been selected manually. Working with the experts from the Turbines Supply Chain Unit, 16 key requirements of the rear lock plate have been identified. The filtered rear lock plate requirements are illustrated in Figure 5.10. As seen in the figure, there are two Non-Functional System Requirements, four requirements in the Key Change Packages section, five Functional Requirements and five Non-Functional Implementation Requirements. Additionally, there are also four Non-Functional Performance Requirements, which are the duplicates of the Functional Requirements. For completeness, the links have been created (blue arrows in Figure 5.10) to signify that these requirements are the copies of the Functional Requirements. However, there should only be the unique requirements and there should be no duplication as mentioned in the International Council on Systems Engineering (INCOSE) document (Walden et al., 2015).

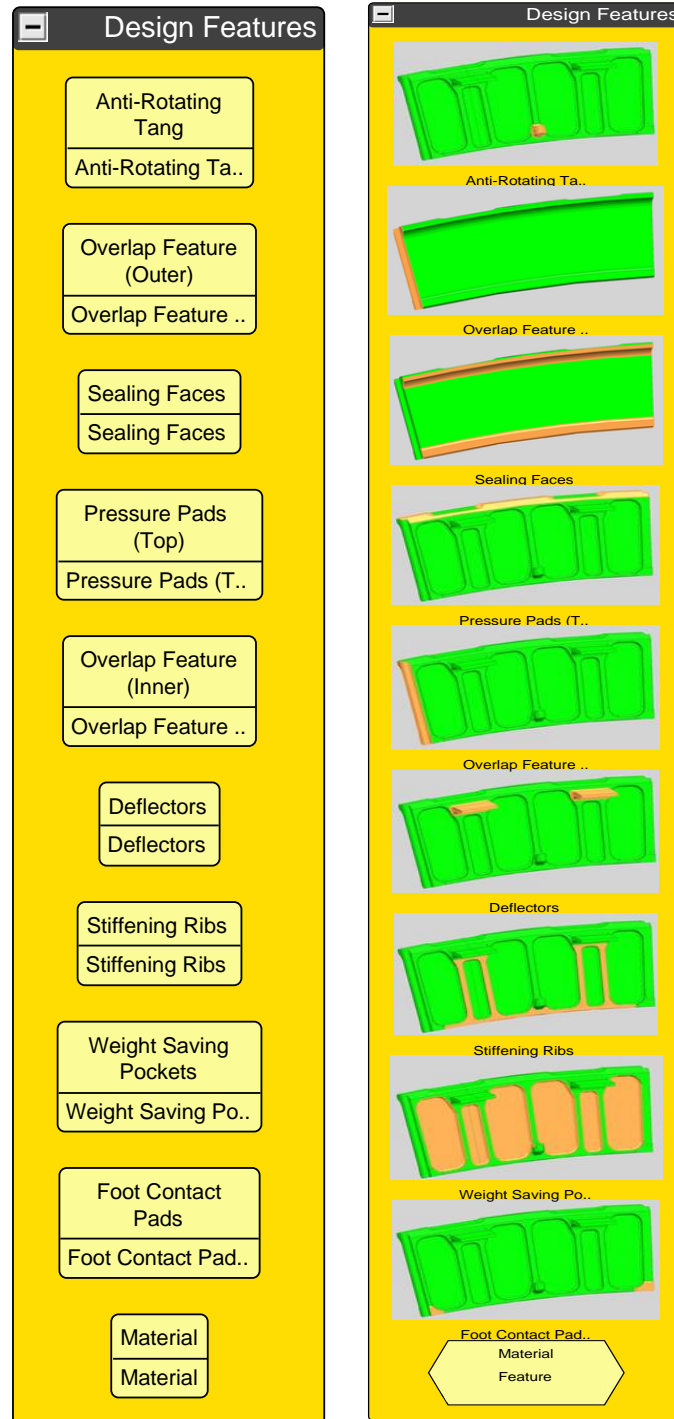


Figure 5.9: The comparison of the Design Features domain with and without the pictures.

Once all three domains have been set, the links have been created between the design features, OP sequences, and the value streams. The links have been created automatically when imported into BOXARR and are shown in Figure 5.11.

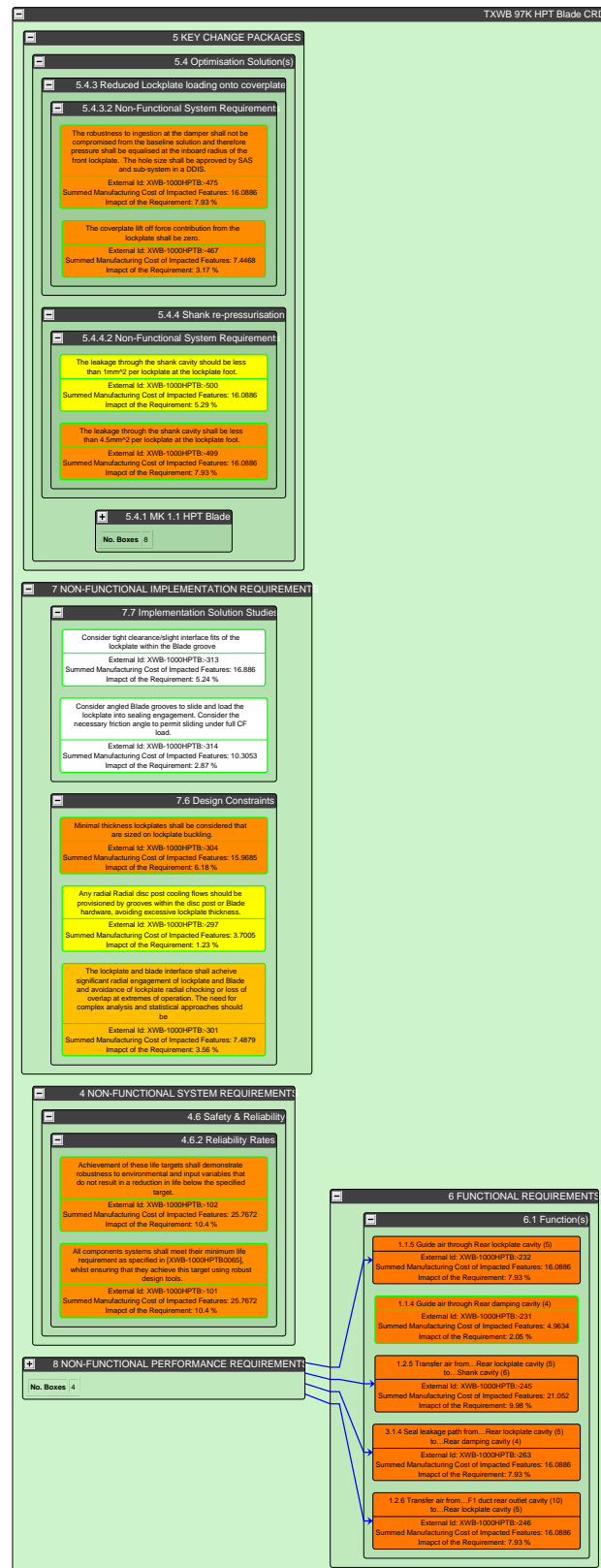


Figure 5.10: The filtered rear lock plate requirements in BOXARR.

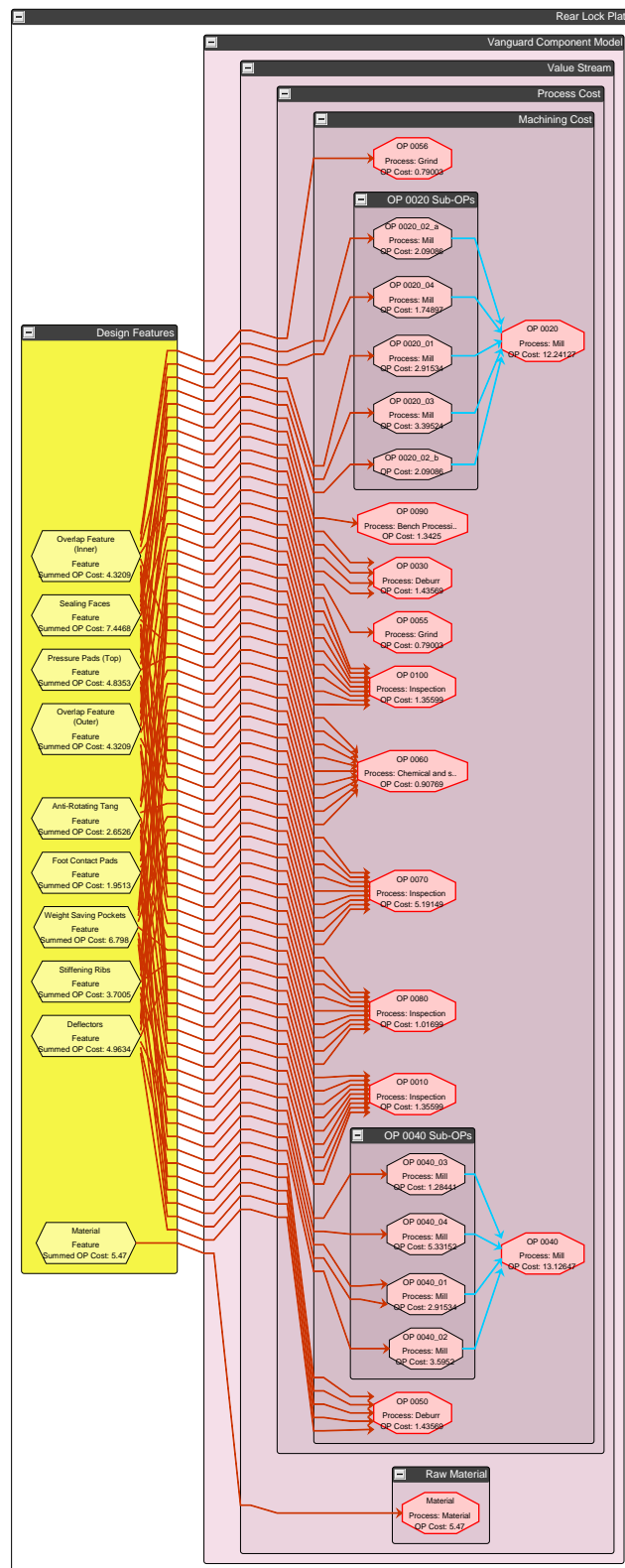


Figure 5.11: The links between the design features and the corresponding operation sequences.

The manufacturing cost of each design feature is also calculated by writing a function that sums the cost of the relevant operation sequences. The value of the summed operation cost is brought forward to the Design Features domain and is seen at the bottom of each design feature box next to the name '*Summed OP Cost*' in Figure 5.11.

The impact is calculated using Equation 4.11. Furthermore, the *Requirements Importance Factor*, *Requirements Link Weight*, *Normalised Feature Cost* and *Feature Link Weight* are calculated using the methods discussed in Section 4.7. The resulting weights of the main variables are summarised in Figure 5.12. As mentioned in Section 4.7, the weights for *Requirements Importance Factor* has been chosen arbitrarily and are summarised in Table 4.1. Additional study has been run to verify the impact of the *Requirements Importance Factor* by varying the weight value. The current weights for modal verbs *MUST*, *SHALL*, as well as all *Functional requirements* are set to 9. To study the impact additional calculations have been done using the weights of 8 and 10. The results are summarised in Figure 5.13. The cells marked in yellow represent the *SHOULD* type requirements and indicate that the weight value of these requirements has been kept the same (i.e. 6 as summarised in Figure 5.12).

Requirements ID	Requirement Importance Factor (R_i)	Requirement Link Weight (R_{cw})
XWB-1000HPTB:-101	9	0.5
XWB-1000HPTB:-102	9	0.5
XWB-1000HPTB:-245	9	0.4
XWB-1000HPTB:-475	9	0.3
XWB-1000HPTB:-232	9	0.3
XWB-1000HPTB:-246	9	0.3
XWB-1000HPTB:-263	9	0.3
XWB-1000HPTB:-499	9	0.3
XWB-1000HPTB:-304	9	0.3
XWB-1000HPTB:-500	6	0.3
XWB-1000HPTB:-313	6	0.4
XWB-1000HPTB:-301	9	0.2
XWB-1000HPTB:-467	9	0.1
XWB-1000HPTB:-314	6	0.2
XWB-1000HPTB:-231	9	0.1
XWB-1000HPTB:-297	6	0.1

Design Feature	Normalised Feature Cost (W_c)	Feature Link Weight (W_f)
Weight Saving Pockets	0.1658	0.0652
Deflectors	0.1211	0.0870
Stiffening Ribs	0.0903	0.0870
Foot Contact Pads	0.0476	0.0217
Pressure Pads (Top)	0.1180	0.1087
Sealing Faces	0.1817	0.1957
Anti-Rotating Tang	0.0647	0.0435
Overlap Feature (Inner)	0.1054	0.1522
Overlap Feature (Outer)	0.1054	0.1522
Material	0.1334	0.0870

Figure 5.12: Summary of the calculated weight factors.

As seen in Figure 5.13, there is a change of 13% in the values of the *Requirement Links Complexity*, *Feature Cost Weight* and *Feature Links Complexity* when increasing the value of the *Requirements Importance Factor* from 8 to 9; and 25% change when increasing from 8 to 10. However, the change in the *Requirements Impact* value is small: 2% when going from weight 8 to 9, and 3% when going from weight 8 to 10. Because Rolls-Royce is already using the weighting system from 1 to 9 in other tools it was decided to use the *Requirements Importance Factor* of 9 to represent the most important requirements.

In order to calculate the impact of each requirement, the links between the requirements and the impacted design features have to be created. This was accomplished with the help of experts from the Turbine SCU by manually linking the identified requirements to the relevant design features. As soon as this step was completed, the impact was instantly calculated. The mapped requirements to design features are shown in Figure

Requirements ID	Requirement Importance Weights									
	8	9	10	8	9	10	8	9	10	8
	Features Link Complexity (F_c)			Features Cost Weight (FC_w)			Requirements Link Complexity (R_c)			Impact (%)
XWB-1000HPTB-101	3.48	3.91	4.35	5.03	5.66	6.29	4.00	4.50	5.00	10.21
XWB-1000HPTB-102	3.48	3.91	4.35	5.03	5.66	6.29	4.00	4.50	5.00	10.21
XWB-1000HPTB-245	4.70	5.28	5.87	4.11	4.62	5.14	3.20	3.60	4.00	9.80
XWB-1000HPTB-475	4.00	4.50	5.00	3.14	3.53	3.93	2.40	2.70	3.00	7.79
XWB-1000HPTB-232	4.00	4.50	5.00	3.14	3.53	3.93	2.40	2.70	3.00	7.79
XWB-1000HPTB-246	4.00	4.50	5.00	3.14	3.53	3.93	2.40	2.70	3.00	7.79
XWB-1000HPTB-263	4.00	4.50	5.00	3.14	3.53	3.93	2.40	2.70	3.00	7.79
XWB-1000HPTB-499	4.00	4.50	5.00	3.14	3.53	3.93	2.40	2.70	3.00	7.79
XWB-1000HPTB-304	1.91	2.15	2.39	3.12	3.51	3.90	2.40	2.70	3.00	6.06
XWB-1000HPTB-500	3.00	3.00	3.00	2.36	2.36	2.36	1.80	1.80	1.80	5.84
XWB-1000HPTB-313	2.22	2.22	2.22	2.47	2.47	2.47	2.40	2.40	2.40	5.79
XWB-1000HPTB-301	1.22	1.37	1.52	1.46	1.64	1.83	1.60	1.80	2.00	3.49
XWB-1000HPTB-467	1.57	1.76	1.96	1.45	1.64	1.82	0.80	0.90	1.00	3.12
XWB-1000HPTB-314	1.17	1.17	1.17	1.51	1.51	1.51	1.20	1.20	1.20	3.17
XWB-1000HPTB-231	0.70	0.78	0.87	0.97	1.09	1.21	0.80	0.90	1.00	2.01
XWB-1000HPTB-297	0.52	0.52	0.52	0.54	0.54	0.54	0.60	0.60	0.60	1.36
										1.23
										1.12

Figure 5.13: The impact of different Requirements Importance Factor weight values.

5.14 and the whole relational map in Figure 5.15. The constructed relational map will serve as a basis for trade-off studies and component cost optimisation activities.

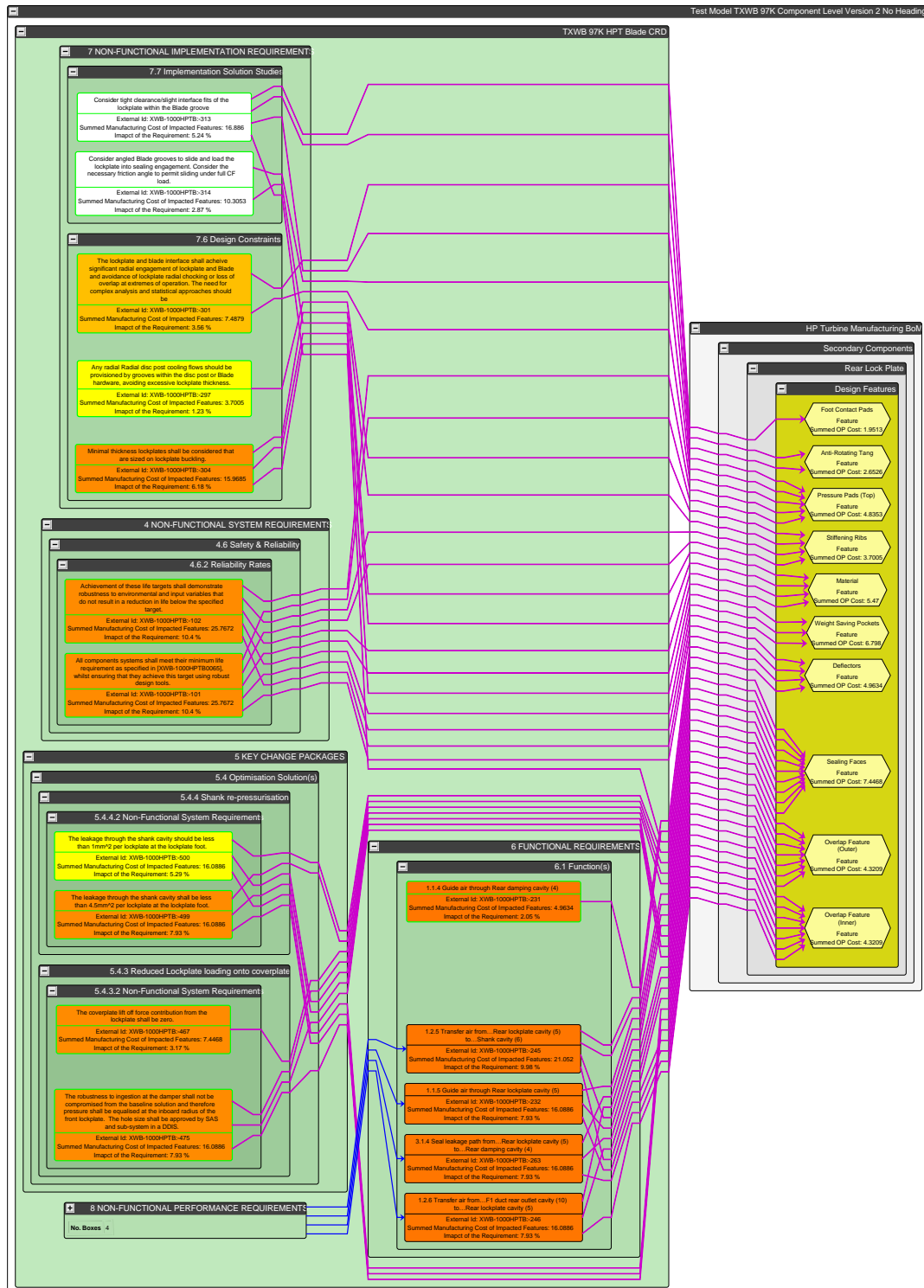


Figure 5.14: The mapped requirements to design features.

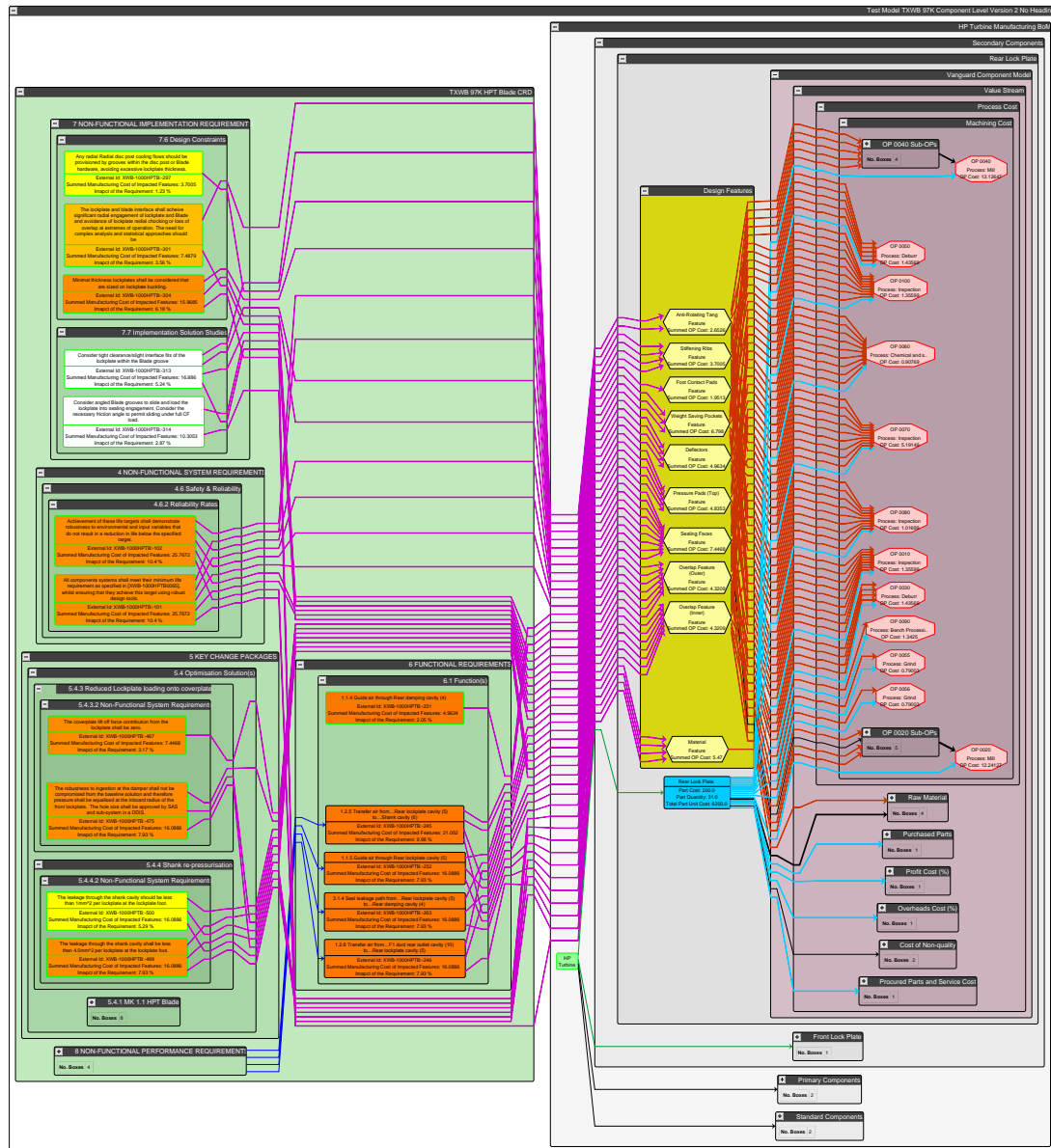


Figure 5.15: The holistic map of the relations between the requirements, design and cost domains.

5.2.1 The Result

The modelled graph, shown in Figure 5.15, can be used to aid the designer or engineer when analysing the requirements relationships to cost and their overall impact at the Component Level. The graph structure is illustrated in Figures 5.16 and 5.17. It is not limited to this particular structure however. The data can be organised and structured following the standards of the individual company. Looking at Figure 5.16, the Trent engine Component Level group contains the HPT Blade CRD group and HP Turbine Manufacturing BoM group. The HPT Blade CRD contains the requirements of the component. The HP Turbine Manufacturing BoM group is comprised of three sub-groups and a box. Following Rolls-Royce's standards, the components in the sub-system are

classified by the component type. In this case there are three component types: Primary Components, Secondary Components and Standard Components. As illustrated in Figure 5.16, primary components are critical to the engine and often most expensive (i.e. HPT Blade, HPT Disc etc.); secondary components are less critical but still required for the engine to function and are often outsourced (i.e. lock plates, seal plates etc.); Standard components are the parts such as nuts and bolts that can be bought off the shelf. The green box calculates the total cost of the HP Turbine sub-system by adding the costs of all the components in the sub-system.

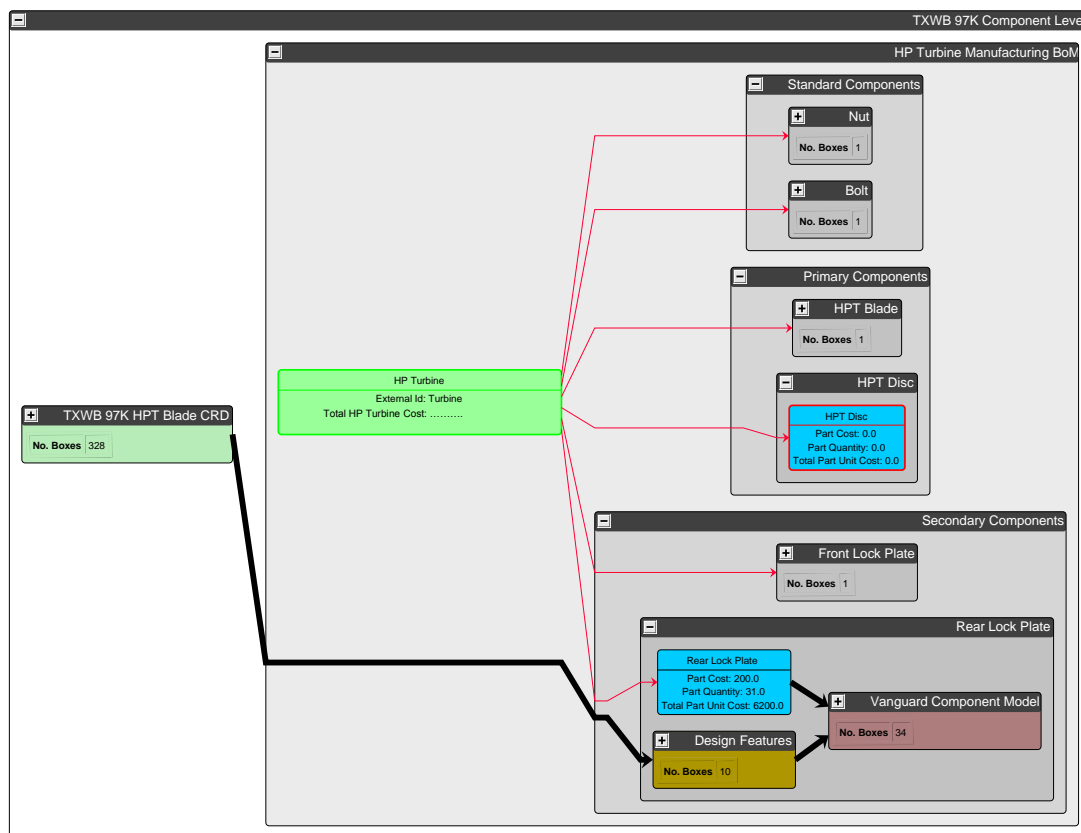


Figure 5.16: The graph structure.

At the component group level (i.e. Rear Lock Plate group) the design features data and cost data is stored. The former resides in the Design Features group and the latter in the Vanguard Component Model group. Moreover, seven value streams reside within the Vanguard Component Model group as depicted in Figure 5.17. The blue box calculates the Part cost, and the total Set cost when the quantity value is provided. The structure in the component group can be replicated for every part across the engine. In other words, each component group is comprised of Design Features domain and Cost domain and can be scaled to any number of parts. Figure 5.18 shows the design features of the Rear Lock Plate and the links to the OP sequences.

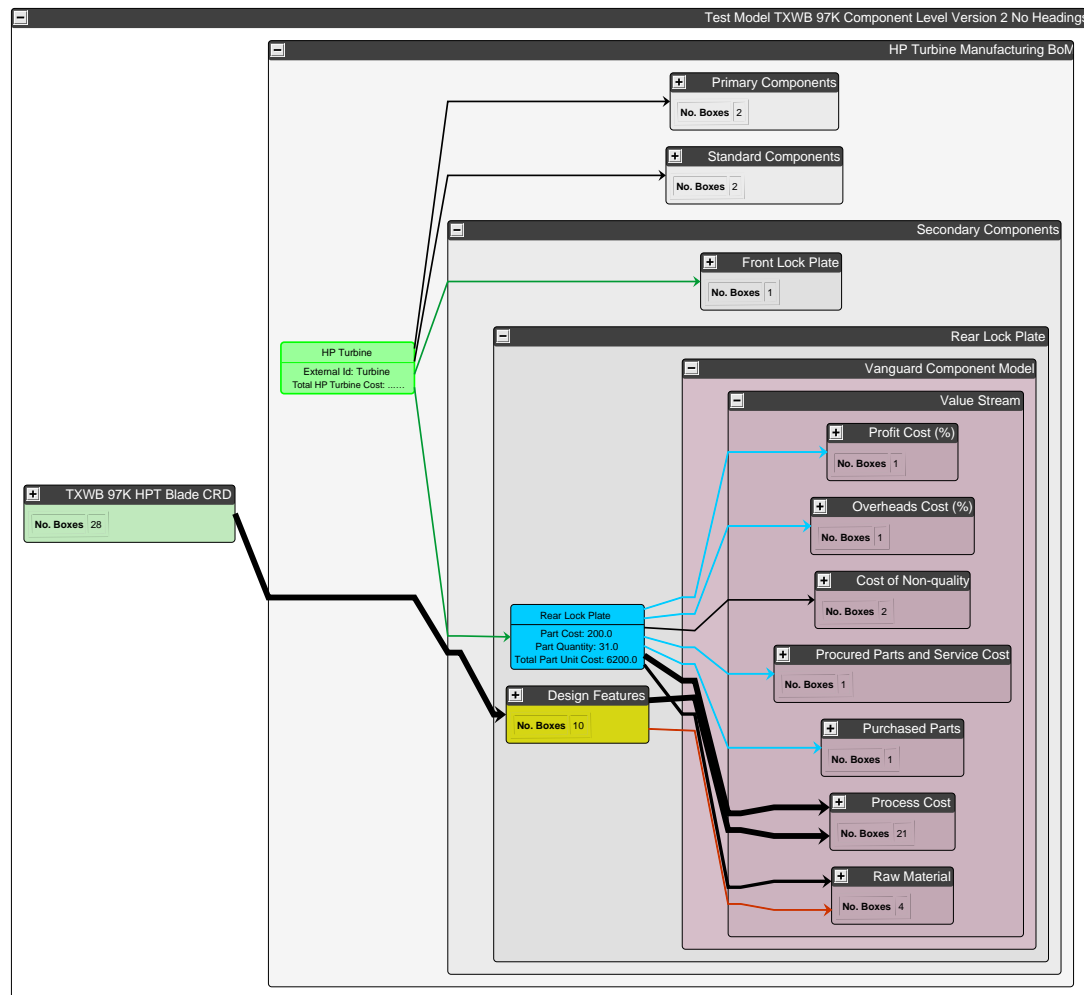


Figure 5.17: The graph structure.

The created links indicate which operation sequences are required in order to manufacture each design feature. The summed OP costs of each feature are displayed in yellow features boxes. One of the goals in this research is to create the links between the design features and cost data in automated way by selecting the relevant data fields from the CSV file where the cost data has been saved. Once the data has been imported, the part cost and the total set cost has been calculated and displayed in the blue box. As seen in Figure 5.16, the cost of the Rear Lock Plate is £200. This amount matches the amount in the Vanguard Model shown in Figure 5.5. It can, therefore, be concluded that the built-in functions are working correctly without any errors.

Having the structure, as shown in 5.16, significantly improves the manageability of the data. Furthermore, the data becomes easily traceable. The filtering feature in BOXARR further adds value when navigating through a complex graph with thousands of nodes (i.e. boxes). The filtering tool plays a crucial role during the analysis of the graph. For example, if the designer wants to see how many features as well as which features are affected by one requirement, he/she only needs to filter the downstream connections of a particular requirement. As illustrated in Figure 5.19, the requirement XWB-

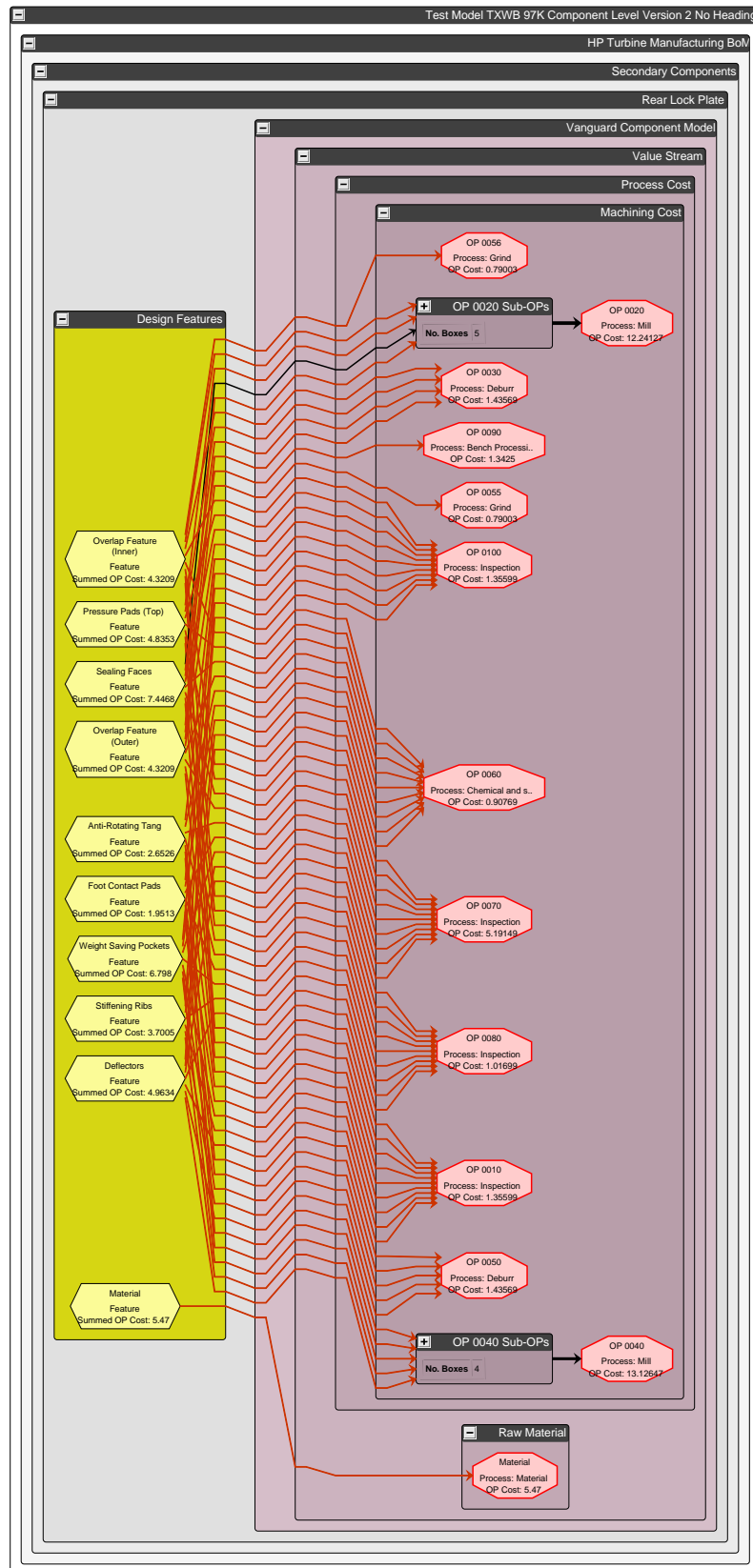


Figure 5.18: Design features of the Rear Lock Plate and the links to the OP sequences.

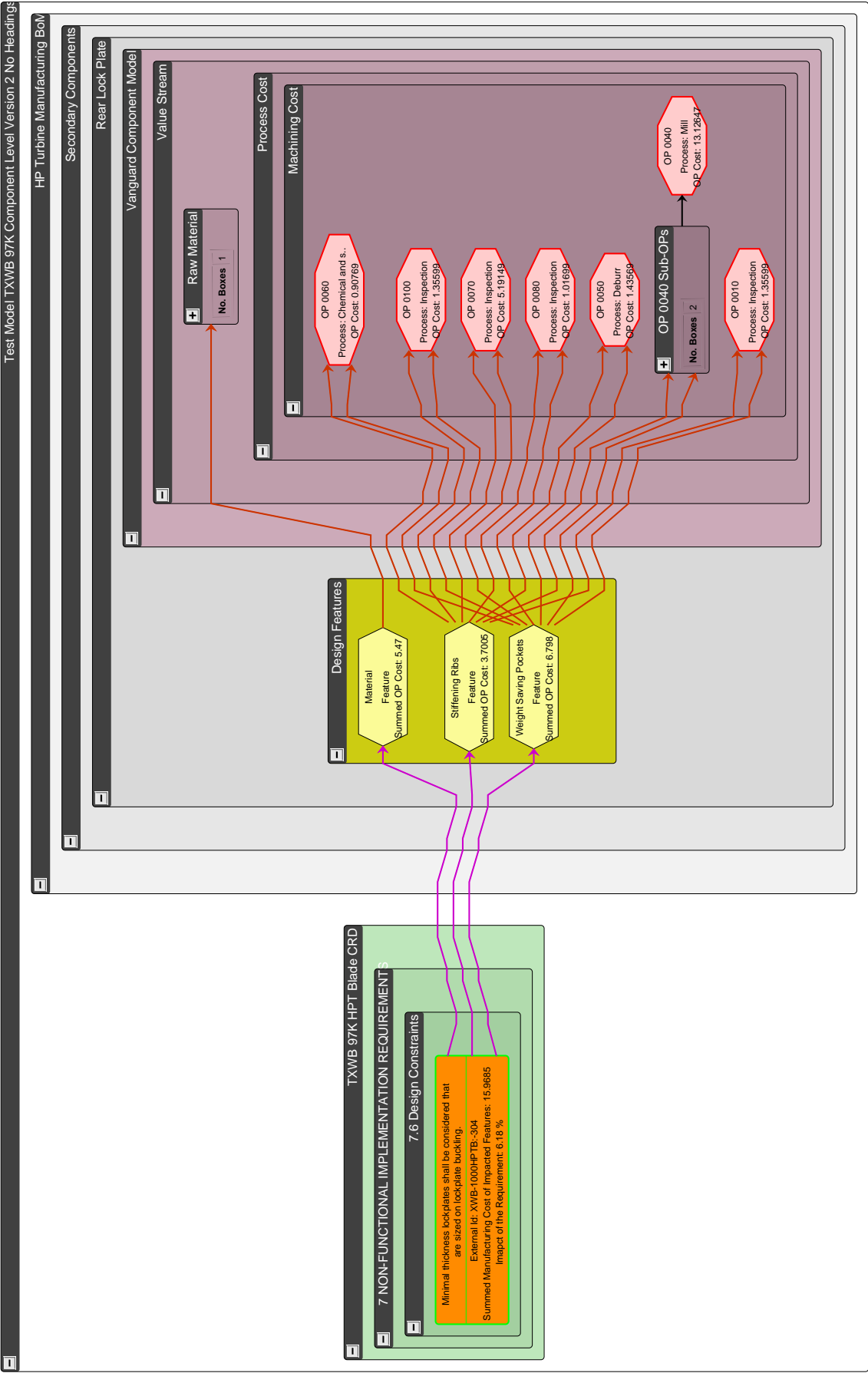


Figure 5.19: Filtered downstream connections of a particular requirement.

1000HPTB:-304 affects three design features (Stiffening Ribs, Weight Saving Pockets and Material). All the links and cost data is clearly visible. To see the graphical representation of the features, only a single click of the button is required. The result is shown in Figure 5.20. Similarly, if the designer wants to see how many requirements affect one feature, he/she needs to filter the upstream and downstream connections of a particular feature as shown in Figure 5.21. Again, it is clearly visible that requirements XWB-1000HPTB:-101, XWB-1000HPTB:-102 and XWB-1000HPTB:-304 affect the '*Weight Saving Pockets*' feature. Additionally, the OP sequences required to manufacture this feature are also shown together with the summed OP sequence cost of the feature. As seen in Figure 5.21, the following OP sequences are required to fabricate the '*Weight Saving Pockets*' feature: OP 0010 Inspection, OP 0040 Mill, OP 0050 Deburr, OP 0060 Chemical and Surface Treatment, OP 0070 Inspection, OP 0080 Inspection, OP 0100 Inspection. By adding the costs of the OP sequences, the manufacturing cost of the Weight Saving Pockets feature comes out as £6.80.

Having the relational graph with the references to cost offers the designer a holistic view of structured data to facilitate informed decision making. For example, a designer can now change the feature to see how it affects the cost, merge the requirements to check the impact on cost or even change the whole manufacturing process, for example comparing a Machining approach to Metal Injection Moulding.

Regulation and competition in the aerospace industry forces companies to concentrate on the safety and performance aspects of the product. Cost considerations are often secondary. The current CAD tools provide the platform to conduct the structural optimisation through FEA analysis, performance optimisation through CFD analysis. However, cost optimisation is not adequately supported. Although tools to manage cost exist, these often come as a standalone applications that have no direct links between cost and design domains. This methodology alongside the BOXARR software establishes a link between the design and cost domains, thus providing a platform for cost optimisation activities during the design process. Another advantage is that the cost data is factual. It can always be updated to the most recent economic year. The model becomes especially useful if an old component is used in a new project with minimal or substantial changes. The old model could be updated to the most recent economic data and used as a starting point in the redesign activities. As a result, the product development time as well as the amount of people working on the component in early design stages can be reduced. This, in turn, is translated into reduced costs.

As previously discussed, the impact of each requirement is calculated instantly by using Equation 4.11 once the link between the requirement and design feature is created. The created links not only signify the relations between two nodes, but also serve as bridges between the domains. Thus allowing the data from cost domain to be brought forward to design domain and requirements domain. As demonstrated by the example in Figure 5.22, requirement XWB-1000HPTB:-304 impacts the cost of the Rear Lock Plate by 6.18%. Additionally, the total process cost of impacted features is also displayed in

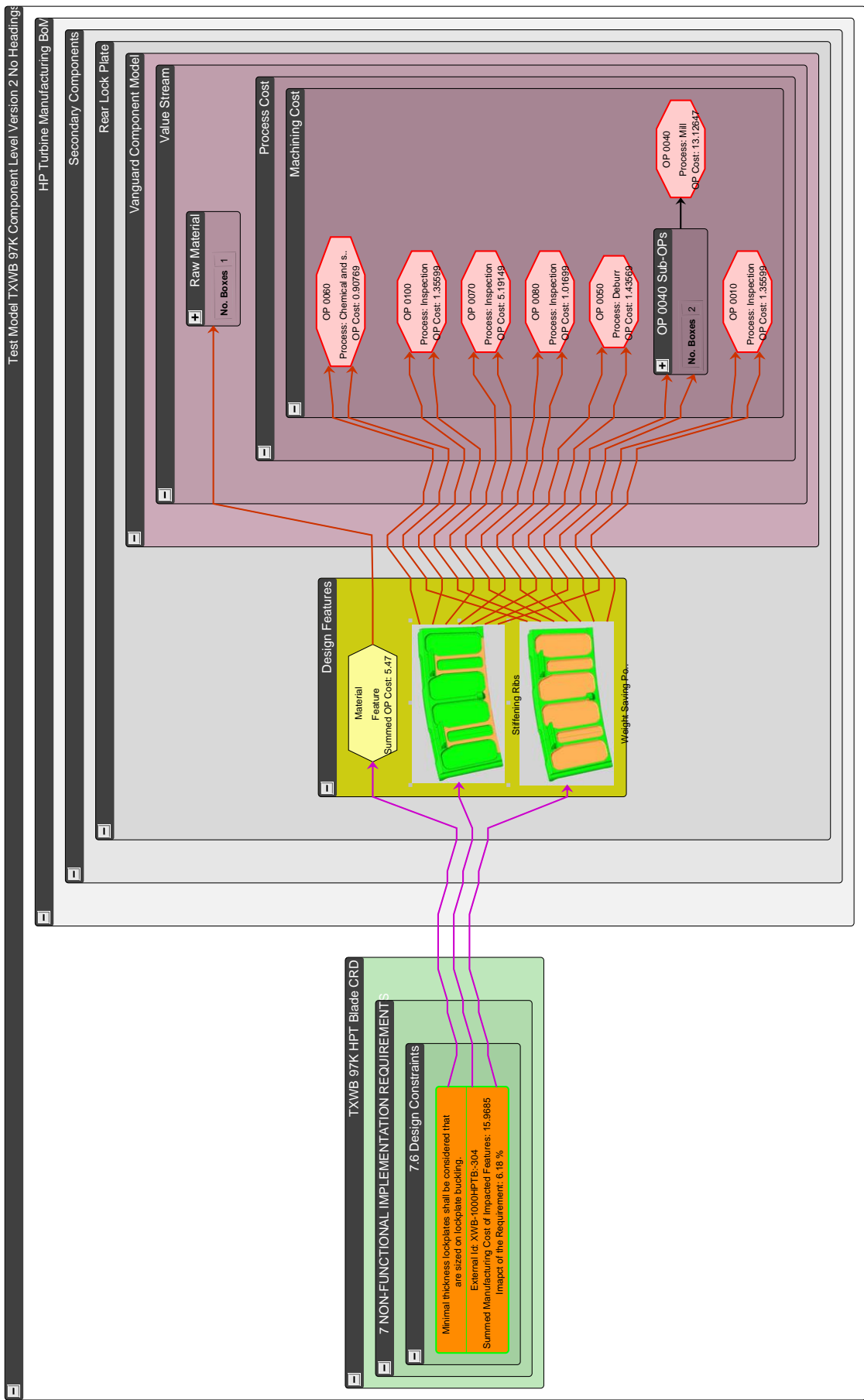


Figure 5.20: Filtered downstream connections of a particular requirement with the graphical representation of features.



the requirements box. The impacts of all 16 requirements, together with the weights and other corresponding parameters are summarised in Figure 5.23. As seen from the figure, requirements XWB-1000HPTB:-101 and XWB-1000HPTB:-102 have the biggest impact. These two requirements are related to safety and reliability.

The benefits of using the proposed decision support methodology are demonstrated in the following section.

5.3 Methodology Demonstration: The Rear Lock Plate Redesign Activity

In this section, the proposed methodology is applied to a real example case in order to highlight its benefits.

The upgraded version of the XWB engine has been planned to be released in 2018 delivering a 1% reduction in fuel consumption. The task to reduce the overall cost of the engine was given to the designers. The rear lock plate was chosen as a candidate component for cost reduction. The cost reduction simulation activity was initiated and carried out by the author with the help of experts. To evaluate the potential areas for cost reduction, relational dependency map, created in BOXARR, was examined. The summary of the requirements impact and the representative manufacturing cost of each feature is presented in Figures 5.23 and 5.24 respectively. The architecture of the engine has not been changed, therefore one of the requirements was to reduce the cost without changing the size of the lock plate or any feature that interfaces with other components.

The relational map of the rear lock plate shown in Figure 5.15 was explored in order to check for dependencies between the requirements and features. The geometry model of the rear lock plate was also evaluated for potential cost reduction areas. Two design features were identified as potential cost reduction targets. After the discussion with the structural and thermal engineers the design feature ‘*Deflectors*’ was chosen to be removed, since the benefit of having this feature had not been measured. According to one of the designers, the ‘*Deflectors*’ feature is a legacy design feature, therefore it has always been reused. The theoretical function of the ‘*Deflectors*’ is to direct the cooling flow towards the disc post. However, the cooling effectiveness is not currently known. Two potential reasons could be identified. First, because of the instrumentation difficulties. Second, the original design intent was verified and understood in theory, however, the design knowledge and rationale have not been captured. ‘*Weight Saving Pockets*’ design feature was chosen as the second cost reduction target. The ‘*Deflectors*’ design feature was removed and the number of ‘*Weight Saving Pockets*’ was reduced from 6 to 2. The new rear lock plate geometry model is shown in Figure 5.25.

To evaluate the cost of the new geometry, the Vanguard cost model was updated. The manufacturing operation that produced the ‘*Deflectors*’ design feature was suppressed

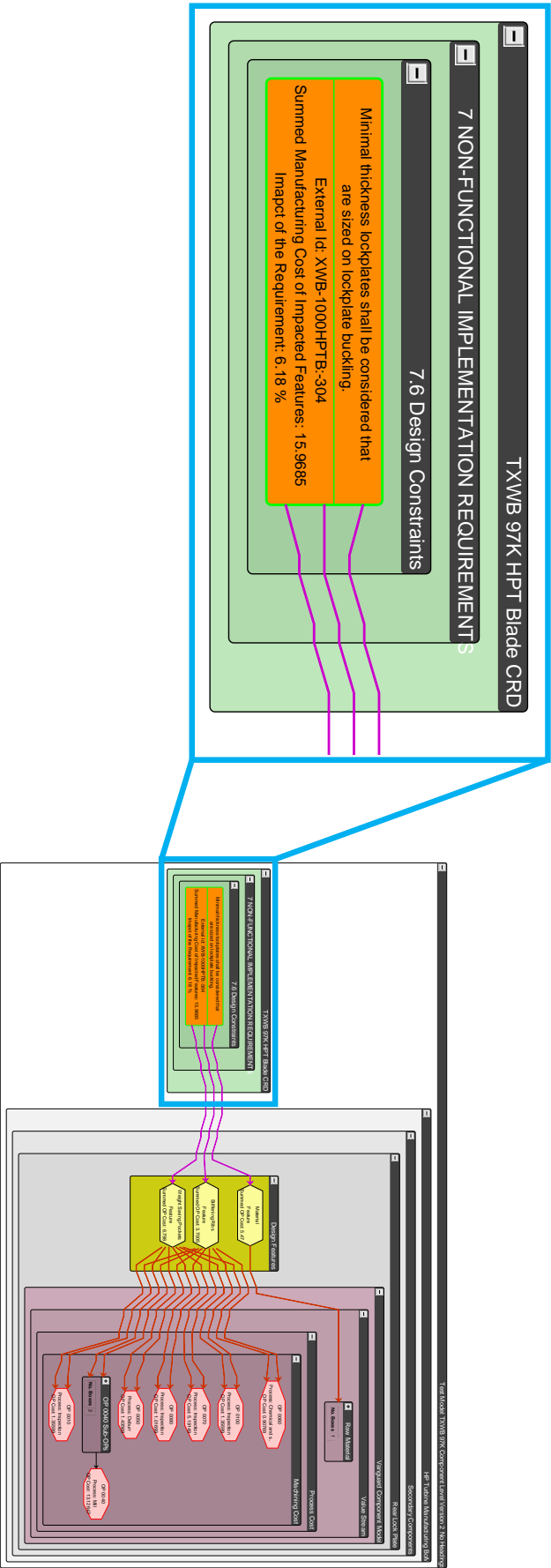


Figure 5.22: Calculated impact of the requirement.

Requirements ID	Number of Links to Features	Features (component it belongs to)	Number of Links to Requirements (from features)	Weights on Features Complexity (F_c)	Normalised Feature Costs	Weights on Features Cost (FC_w)	Weights on Requirements Complexity (R_c)	Impact
XWB-1000HPTB:-101	5	Weight Saving Pockets	3	3.91	0.17	5.66	4.50	10.40%
		Deflectors	4		0.12			
		Stiffening Ribs	4		0.09			
		Pressure Pads (Top)	5		0.12			
		Material	4		0.13			
XWB-1000HPTB:-102	5	Weight Saving Pockets	3	3.91	0.17	5.66	4.50	10.40%
		Deflectors	4		0.12			
		Stiffening Ribs	4		0.09			
		Pressure Pads (Top)	5		0.12			
		Material	4		0.13			
XWB-1000HPTB:-245	4	Deflectors	4	5.28	0.12	4.62	3.60	9.98%
		Sealing Faces	9		0.18			
		Overlap Feature (Inner)	7		0.11			
XWB-1000HPTB:-475	3	Overlap Feature (Outer)	7	4.50	0.11	3.53	2.70	7.93%
		Sealing Faces	9		0.18			
		Overlap Feature (Inner)	7		0.11			
		Overlap Feature (Outer)	7		0.11			
		Sealing Faces	9		0.18			
XWB-1000HPTB:-232	3	Overlap Feature (Inner)	7	4.50	0.11	3.53	2.70	7.93%
		Overlap Feature (Outer)	7		0.11			
		Sealing Faces	9		0.18			
XWB-1000HPTB:-246	3	Overlap Feature (Inner)	7	4.50	0.11	3.53	2.70	7.93%
		Overlap Feature (Outer)	7		0.11			
		Sealing Faces	9		0.18			
XWB-1000HPTB:-263	3	Overlap Feature (Inner)	7	4.50	0.11	3.53	2.70	7.93%
		Overlap Feature (Outer)	7		0.11			
		Sealing Faces	9		0.18			
XWB-1000HPTB:-499	3	Overlap Feature (Inner)	7	4.50	0.11	3.53	2.70	7.93%
		Overlap Feature (Outer)	7		0.11			
		Sealing Faces	9		0.18			
XWB-1000HPTB:-304	3	Weight Saving Pockets	3	2.15	0.17	3.51	2.70	6.18%
		Stiffening Ribs	4		0.09			
		Material	4		0.13			
XWB-1000HPTB:-500	3	Sealing Faces	9	3.00	0.18	2.36	1.80	5.29%
		Overlap Feature (Inner)	7		0.11			
		Overlap Feature (Outer)	7		0.11			
XWB-1000HPTB:-313	4	Foot Contact Pads	1	2.22	0.05	2.47	2.40	5.24%
		Pressure Pads (Top)	5		0.12			
		Sealing Faces	9		0.18			
		Anti-Rotating Tang	2		0.06			
		Pressure Pads (Top)	5		0.12			
XWB-1000HPTB:-301	2	Anti-Rotating Tang	2	1.37	0.06	1.64	1.80	3.56%
		Pressure Pads (Top)	5		0.12			
		Material	4		0.13			
XWB-1000HPTB:-467	1	Sealing Faces	9	1.76	0.18	1.64	0.90	3.17%
		Pressure Pads (Top)	5		0.12			
		Material	4		0.13			
XWB-1000HPTB:-314	2	Pressure Pads (Top)	5	1.17	0.12	1.51	1.20	2.87%
		Material	4		0.13			
		Deflectors	4		0.12			
XWB-1000HPTB:-231	1	Deflectors	4	0.78	0.12	1.09	0.90	2.05%
		Stiffening Ribs	4		0.09			
XWB-1000HPTB:-297	1	Stiffening Ribs	4	0.52	0.09	0.54	0.60	1.23%

Figure 5.23: Summarised impacts of all 16 requirements.

Feature	Manufacturing Cost of the Feature (in £)
Sealing Faces	7.45
Weight Saving Pockets	6.80
Material	5.47
Deflectors	4.96
Pressure Pads (Top)	4.84
Overlap Feature (Outer)	4.32
Overlap Feature (Inner)	4.32
Stiffening Ribs	3.70
Anti-Rotating Tang	2.65
Foot Contact Pads	1.95

Figure 5.24: Representative features manufacturing cost of the rear lock plate.

and the number of pockets was changed from six to two in the Vanguard cost model. The new cost data was imported into BOXARR to calculate the impact of each requirement and to capture the new relational knowledge between the domains for future reuse. Because the ‘*Deflectors*’ design feature was deleted, the links between ‘*Deflectors*’ and relevant requirements were deleted automatically when new data was imported into BOXARR. The summary of the requirements impact and the manufacturing cost of each feature of the new rear lock plate model is presented in Figures 5.26 and 5.27 respectively.

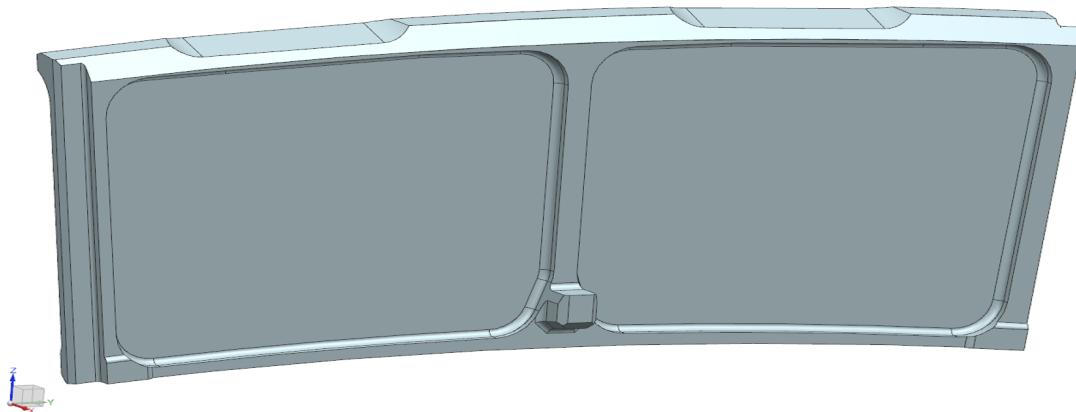


Figure 5.25: The CAD model of the redesigned rear lock plate.

The relational dependency graph was very useful during the selection process of the potential cost reduction areas. The established links between the requirements, design, and cost domains provided the meaningful insight into the design of the rear lock plate. Following the relational links between the domains, features that satisfy the particular requirements were easy to identify. Furthermore, the links to the manufacturing cost of

each feature were clearly visible. As seen in Figure 5.24, the manufacturing costs of the ‘Deflector’ and ‘Weight Saving Pockets’ features were £4.96 and £6.80 respectively. It is the second and third highest manufacturing cost in the table excluding material cost.

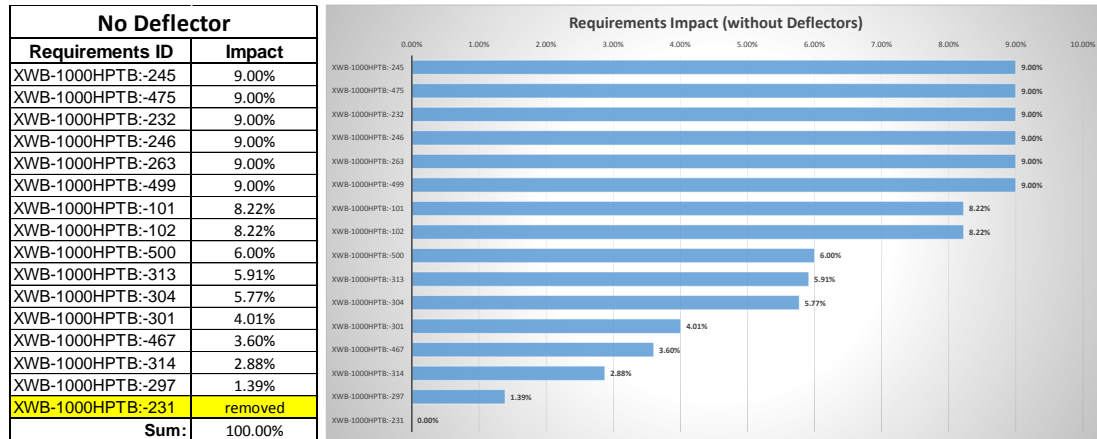


Figure 5.26: Requirements impact summary of the new rear lock plate model.

Feature	Manufacturing Cost of the Feature
Sealing Faces	7.59
Weight Saving Pockets	3.43
Material	2.80
Deflectors	removed
Pressure Pads (Top)	4.98
Overlap Feature (Outer)	4.61
Overlap Feature (Inner)	4.61
Stiffening Ribs	3.92
Anti-Rotating Tang	2.87
Foot Contact Pads	2.17

Figure 5.27: Representative manufacturing cost of each feature of the new rear lock plate model.

The fact that manufacturing costs were easily identifiable and the links to design features were apparent, helped to determine the candidate features for cost reduction activities relatively quickly. If this tool had not been present, to find out the costs of each feature would have taken much longer. When investigating the dependency links between the requirements, it was discovered that one of the requirements was redundant and could be merged with another requirement. Requirement *XWB-1000HPTB:-231* ‘1.1.4 Guide air through Rear damping cavity (4)’ was satisfied only by the ‘Deflectors’ design feature. Since this feature had been removed, this requirement was merged with requirement *XWB-1000HPTB:-232* ‘1.1.5 Guide air through Rear lock plate cavity (5)’. Again, not having the relational dependency graph between the requirements and design features,

would take longer to establish the relational links between them. Using the BOXARR tool the analysis of selecting the candidate areas for cost reduction was completed in less than an hour. The whole redesign activity for cost reduction, including the modification of the CAD model and discussion with the experts, but excluding the FEA and CFD analysis, took less than a full working day. It should be noted, however, that the majority of time was spent waiting for the experts to become available. Since the rear lock plate was one of the important components of the Turbine Sub-System, the author had to consult with experts due to his limited experience and knowledge about the component. The value stream costs (of the Rear Lock Plate with and without the ‘*Deflectors*’) are compared in Figure 5.28. The new results demonstrate that the overall component cost has been reduced by 6%.

Value Stream	Rear Lock Plate With Deflectors Cost (£)	Rear Lock Plate Without Deflectors Cost (£)	Cost Reduction
Procured Parts Cost	122.17	120.98	0.98%
Process Cost	40.99	34.19	16.59%
Overheads Cost	26.08	24.88	4.60%
Material	5.47	2.80	48.73%
Cost of Non-quality	5.29	4.96	6.24%
Total Unit Cost:	£200.00	£187.81	6.09%

Figure 5.28: The comparison of the representative value streams of both rear lock plates with and without the ‘*Deflectors*’.

The fact that the ‘*Deflectors*’ design feature was removed and the number of ‘*Weight Saving Pockets*’ was reduced, means less material was needed, thus leading to reduced casting size. The Material cost was reduced by 48.7%. In addition, the number of manufacturing operations as well as manufacturing process time was reduced as a consequence of the redesign. The reduction of 16.5% in Process cost was achieved. Due to the fact that the most important design data (i.e. requirements, design features, and component unit cost data) was previously captured in the relational dependency graph between the design domains, the cost reduction scenario was simulated in a relatively short period of time. It is interesting to note, however, that the impact of each requirement, illustrated in Figure 5.26, is distributed more evenly for the redesigned part. If the impact of one of the requirements increased as a result of the redesign, it would mean that changes made were a cause of this. This would signal that further analysis was required to investigate the cause of increase. If the cause was justifiable after the analysis, the cost reduction activities would be implemented and requirements impact data stored for future reuse, otherwise different simulation scenarios should be investigated. The change in the requirements impact and its effects are explored in Section 6.3. The feedback received from the experts was that the tool looked promising and the methodology could potentially be implemented in the design process. However, the requirements impact values were judged with caution. More data from several different components is needed to assess the validity of the requirements impact values.

5.4 Discussion

The proposed methodology together with the modelled graph provides a holistic view of the relational dependencies between the requirements and how they are linked to manufacturing process costs. The established links between the requirements domain, design domain and cost domain serve as carriers of information. This, in turn, enables the information from cost domain to be brought forward to requirements domain. BOXARR's graph modelling platform together with the computational functionality provide the means to run '*what if*' scenarios and simulations more efficiently during the trade-off studies and cost optimisation activities. It is safe to say that both the methodology and BOXARR software fall into the generic tools category. Any data saved in a spreadsheet is suitable for use in BOXARR. In addition, BOXARR is flexible and highly customisable and does not bound the user to use a particular modelling format. The methodology, on the other hand, follows the product development life cycle. It provides a platform for analysis in the early design stages. The analysis begins in the requirements stage, then it goes into design and manufacturing stages where the links to cost are established. Once the links are established, the manufacturing cost data of the features that satisfy particular requirement can be brought forward to the requirements domain. In fact, any cost data, that is of interest, can be brought into the requirements domain, be it the total part cost, the summed up operations cost, the feature cost or other costs. The code needs to be written only once in order to display the required information. Furthermore, the proposed methodology captures knowledge. Approximately 90% of new designs are based on existing designs (Kim et al., 2007) thus the modelled relational graph can be used as a starting point in redesign activities as demonstrated in the previous section. The fact that requirements, design features and cost information is in one place saves tremendous amounts of time for new projects. It eliminates the need to go through many documents to search for relevant information and, most importantly, it provides the requirements interdependency graph with the references to cost data.

5.5 Lessons Learned

In the first methodology iteration, the decision was taken to explore the current tools used in each of the four domains, to understand how easy it is to automate the processes in the framework and also to analyse what is required to achieve the automated approach. The following stages (in bold font) have been automated in the first iteration of the methodology design:

- Stage 1 - automated data capture into structured CSV file format was achieved for **requirements** and **cost data**. However, tagging the design features to the relevant operations has been done manually by typing the name of the feature into each operation node in Vanguard. Therefore, further investigation of options

how the cost model creation time could be reduced was needed. In addition, the ‘Design Features’ domain in BOXARR was also created manually. As mentioned in Section 3.2, it is not a standard practice at Rolls-Royce to have a standardised Features data base stored in PLM. Therefore, the options how to automate features extraction have been investigated in the second methodology iteration and has been presented in Chapter 6.

- Stage 2 - data upload into a BOXARR software. This is currently a manual task due to the constraints of BOXARR.
- **Stage 3** - the automated requirements clustering has been achieved through BOXARR.
- **Stage 4** - setting the importance factor to each requirement has been automated in BOXARR.
- Stage 5 - mapping requirements to design features was done manually. The automation aspect using the DFMEA tool has been explored and presented in Chapter 6.
- **Stage 6** - links between the design features and manufacturing operations have been created automatically through BOXARR using the data generated from the cost models.
- **Stage 7** - requirements impact has been calculated automatically in BOXARR.

In addition to the remarks above, the ‘*material*’ is not a *feature* (was used as a feature in the Rear Lock Plate example test case) but a *value stream* and should belong to the cost domain rather than the design domain. After some discussions with designers and engineers it was suggested that knowing the material cost contribution of a particular feature would add a level of detail and would help in the design trade off studies. For example, the material stock for condition of supply or black forging geometry could be optimised by trading between manufacturability and cost. This improvement has been explored and presented in Chapter 6.

Chapter 6

Example Test Cases: IPT Stub Shaft and Triple Seal Flange Joint

6.1 Methodology Improvements

In addition to the Rear Lock Plate example, demonstrated in Chapter 5, the design knowledge capture of a more complex part has been investigated and some further methodology improvements have been made. This chapter summarises the methodology improvements and additional methodology features that have been added using IPT Stub Shaft as a test case.

The *Lessons Learned* section in Chapter 5 noted the areas where further methodology improvements were necessary in order to achieve the goals set out in Section 1.3. The following improvements have been made:

- The design features can be captured automatically from the NX CAD.
- The time to tag the design features in the cost model has been significantly reduced by introducing the feature options selection box. The click of a button tags the feature eliminating the typing element completely.
- The DFMEA tool has been utilised to generate the data that is necessary for automated link creation between functional requirements and cost.
- Additional methodology feature that calculates the material cost of the design feature has been introduced.

Each of the improvements are described in more detail in the following sections.

6.1.1 Automated Design Features Extraction from CAD

The proposed solution how to standardise the Feature names is based on the findings from Section 3.2. One of the findings stated that only a few modelling features had meaningful names assigned to them. Therefore, it was hard to understand which design features had been modelled and what modelling features (in the modelling history tree)

were necessary to produce the design feature. On the other hand, specifying the design features that best describe their function will significantly enhance the knowledge about the part, enabling a better understanding about the design intent as a result. Therefore, the proposed solution is to name the features in a standard/common name. The component feature names should be agreed and signed off by a chief design engineer and stored in the PLM tool for reference. When a component is designed in a CAD package, standard feature names should be used. Furthermore, modelling features should be grouped to define the design feature. Two immediate benefits can be identified. First, the modelling tree becomes more organised with the clear semantic information about the design feature. Second, less time is spent identifying which modelling feature is a part of the design feature. Figure 6.1 shows the selected design feature ‘*Taper Bolt Holes*’ and all the modelling features (in pink) that are necessary to create the design feature. Moreover, when the design feature is selected it is clearly referenced on the geometry model (‘*Taper Bolt Holes*’ highlighted in red).

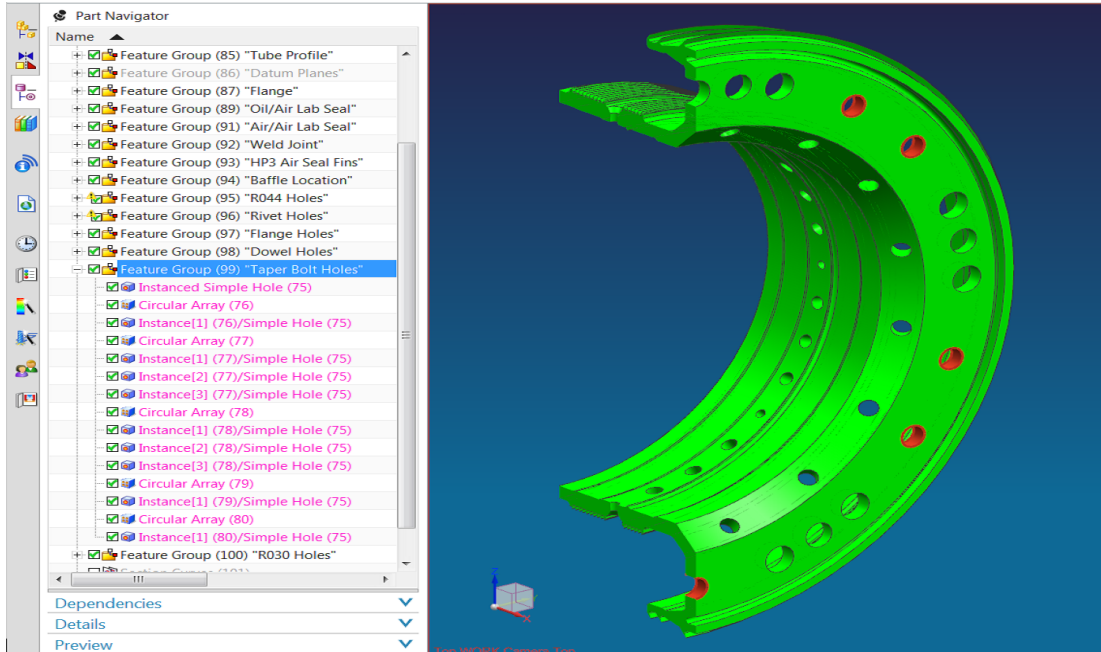


Figure 6.1: Design Feature grouping.

It can be concluded that the design feature definition is a by product of grouping the modelling features. The feature geometry definition is complete only when the standard feature name is assigned to the group. There are two approaches to define the feature geometry. First, by grouping (and using the standard feature name) the relevant sequences of modelling features that define the different design features. Second, by using the first approach and then converting each feature into ‘*Extracted Body*’ (i.e. solid body) and storing in the group with a name *Design Features*. The latter approach has been used in this research. The second approach can be automated by writing a code that converts the design features into the solid bodies and saves them into the *Design*

Features group. However, due to time constraints this step was accomplished manually. The second approach is also beneficial for the feature material mass calculation which is described in Section 6.1.4. The feature geometry can be stored within the part model. Figure 6.2 illustrates the exploded view of the component features and Figure 6.3 the Design Features group in the part file.

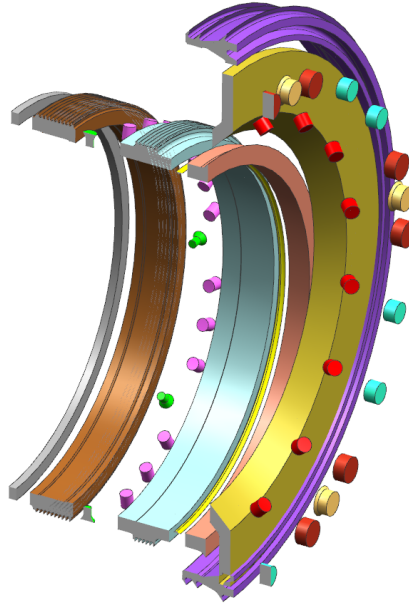


Figure 6.2: IPT Stub Shaft Design Features. Coloured names represent the stub shaft features - HP3 Air Seal Fins, Flange Holes, Taper Bolt Holes, Dowel Holes, R030 Holes, Mating Face, Seal Ring Mating Face, Air/Air Lab Seal, Rivet Holes, R044 Holes, Oil/Air Lab Seal, Weld Joint.

The code in Appendix B extracts the feature names and saves them into the tabular data file. This data is then imported into BOXARR where the design domain with the features is automatically created.

In summary, by grouping the modelling features the design feature is defined as a result. In turn, when all the features are defined the full component geometry is created. Having the component geometry split into features enables the automated extraction of the standard feature names.

6.1.2 Semi-automated Tagging of the Design Features in Vanguard Cost Model

The proposed standard feature definition approach, described in the previous section, has been utilised during the cost model creation. Having a single source of truth (i.e. the standard feature names) it became possible to semi-automate the feature tagging process in the cost model. Rather than typing feature names manually, in the relevant cost

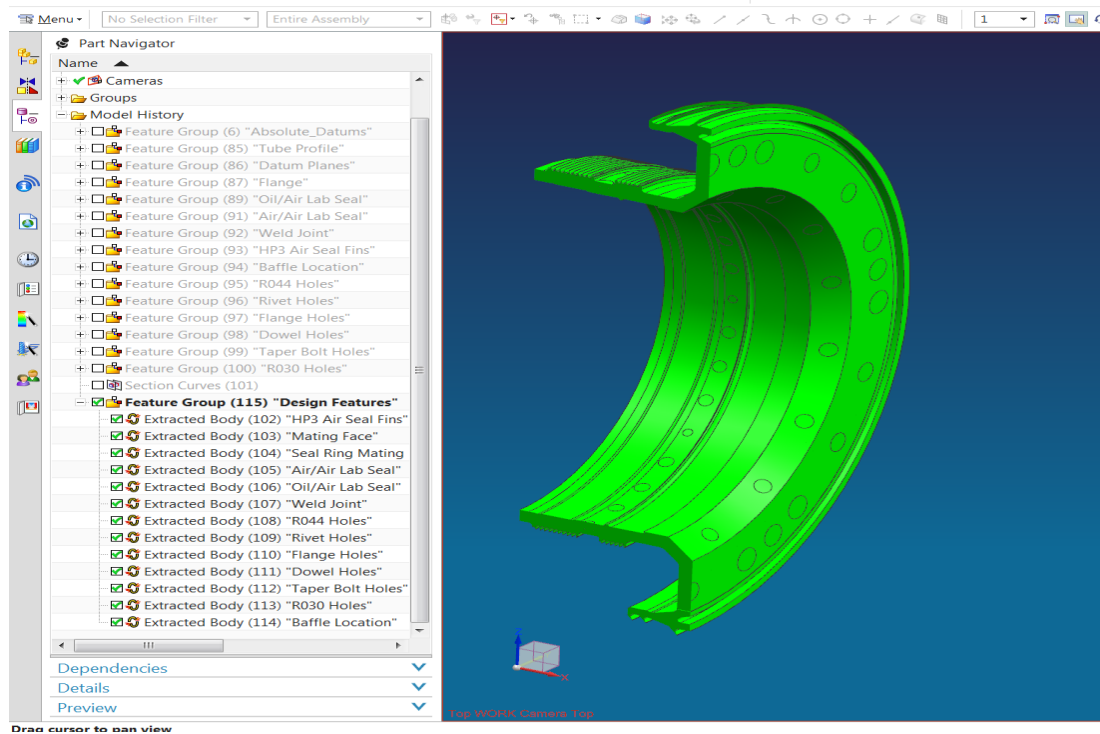


Figure 6.3: Design Features group.

model nodes, this has been accomplished programmatically by providing the capability to select features from the list. Once the feature has been selected for the operation or sub-operation by a cost modeller it was placed in the correct node within the cost model automatically. The functions are generic and can be used for any component cost model. Once these are embedded in the cost model it can be used many times. The addition of this capability resulted in the significant cost model build time reduction. Figure 6.4 illustrates the *Mating Face* feature that is linked to the two grinding sub-operations (highlighted by a red square).

6.1.3 Automated Link Creation between the Requirements and Design Features Using Data from the DFMEA Tool

Design Failure Mode and Effect Analysis tool is an excel tool built by Rolls-Royce and is used to identify all possible failures in a design. The tool also captures knowledge about the risks, potential failures and mitigating actions. As illustrated in Figure 6.5 Failure Mode and Effect Analysis is performed at different stages in the product development lifecycle. It begins in the requirements domain and continues throughout the life of the product or service. The FMEA approach is generic, however, there are variations depending on the domain in which the analysis is performed. There are three major types of the FMEA: Functional Failure Mode and Effect Analysis (FFMEA); Design Failure Mode and Effect Analysis (DFMEA); and Process Failure Mode and Effect Analysis

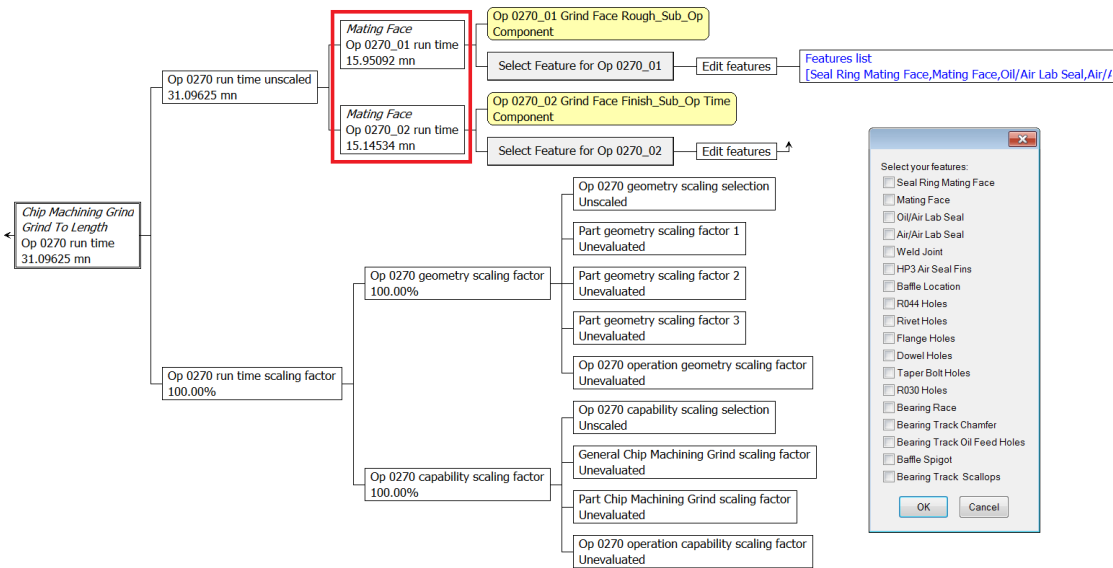


Figure 6.4: Features option list. The red square indicates that *Mating Face* feature is ground. The OP cost is representative.

(PFMEA). As the names suggest FFMEA deals with function failures; DFMEA - with design feature failures; and PFMEA - with process failures. In order to automatically create the links between the requirements and design features a tool that processes the design data as well as overlaps with the requirements domain was required. The DFMEA tool was identified as the best candidate. It was mainly selected for two reasons. First, it captures features information. Second, it also describes the function of the feature. The screen shot of the Rolls-Royce DFMEA excel tool is illustrated in Figure 6.6.

The initial idea was to use the information in the second (Part/Feature) and third (Function) columns (in Figure 6.6) as the data sources to create the links between them automatically. However, the described feature functions were not matching the requirements in the CRD. In addition, any reference to functional requirements was also missing. To overcome this issue the following improvement has been proposed: to add the capability that enables to link the feature to the requirement. The solution was achieved by adding two additional columns for *Requirements ID* and *Requirements Description* which would establish the reference to the requirements. DFMEA is a very time consuming process. It often involves many stakeholders and all the fields are populated manually by typing in text or numbers. To compensate for the manual work that would be required for two extra columns, the following steps were implemented. The VBScript (Appendix D) has been written that imports the features and requirements data from the component features and requirements CSV files that have been created earlier in the process. Once the data has been imported the relevant feature and requirement can be selected using 'Data Validation' function in excel. Furthermore, either *Requirement ID* or *Requirement Description* needs to be selected the rest of the data is populated automatically. For

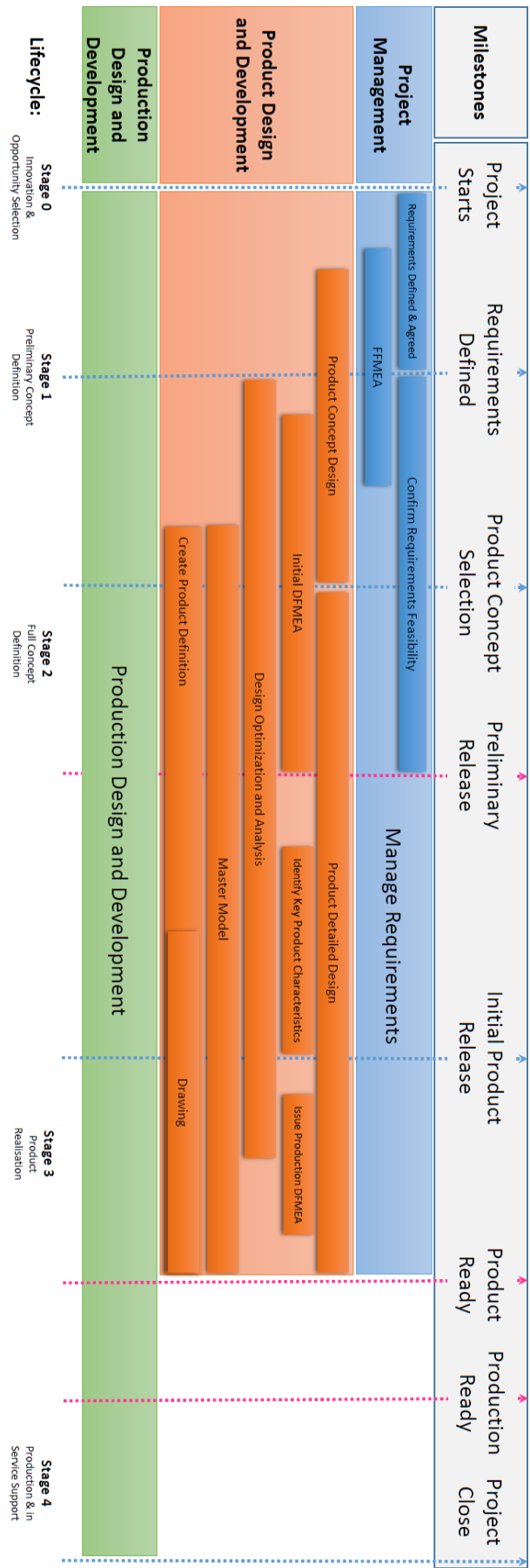


Figure 6.5: Product development lifecycle. The FMEA is performed at the start of the project in the requirements domain; Initial DFMEA - at Stage 1 (Preliminary concept definition) and Production DFMEA - at the end of Stage 3 (Product realisation). Although it is not shown in the Figure the FMEA is performed at the beginning of Stage 3 before the start of production.

6.1.4 Material Cost of the Feature

In the Rear Lock Plate example test case in Chapter 5 the *Material* was identified as a feature by the engineers. It is a design and cost driver. Therefore, it was important for them to capture the cost of the material. However, material cost is a value stream and should be located in the cost domain rather than in the design domain. The following improvement has been implemented: material cost has now been attributed to the feature. It has to be noted, however, that not all features have the material cost attributed to them. In other words, the material cost is attributed only to the features that are solid objects. And any feature that is produced by a material removal method is always located in a solid feature. The argument is that solid features will always be produced first, hence the incurred material cost. For example, the oil hole, as illustrated in Figure 6.8, is located in the bearing race feature, thus the material cost is attributed to the bearing race feature. The amount of material required to produce the feature should be captured from the geometry that represents the initial manufacturing state of component, such as forging, casting or a block/bar of metal. This can be achieved by defining the geometry of the feature (in the condition of supply part) using the method described in Section 6.1.1. An example is demonstrated in Figure 6.9.

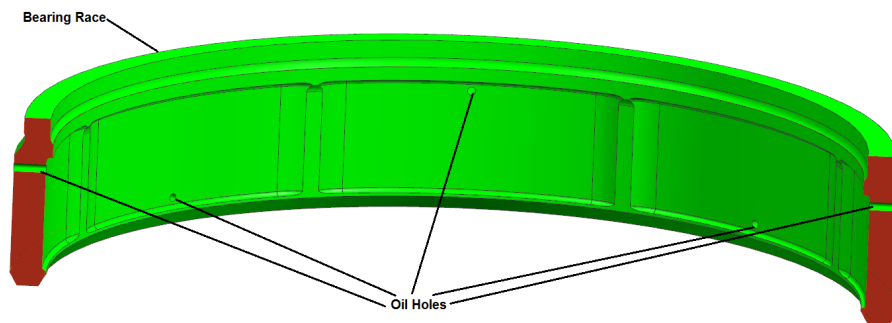


Figure 6.8: Bearing race.

Having the geometry that is split in features enables to measure the volume of material of each feature. The code (Appendix F) has been written to extract the feature material mass ratio from the Condition of Supply (COS) part geometry in CAD, as shown in Figure 6.9 . The ratio is then applied to total material cost in BOXARR to calculate the material cost of each feature.

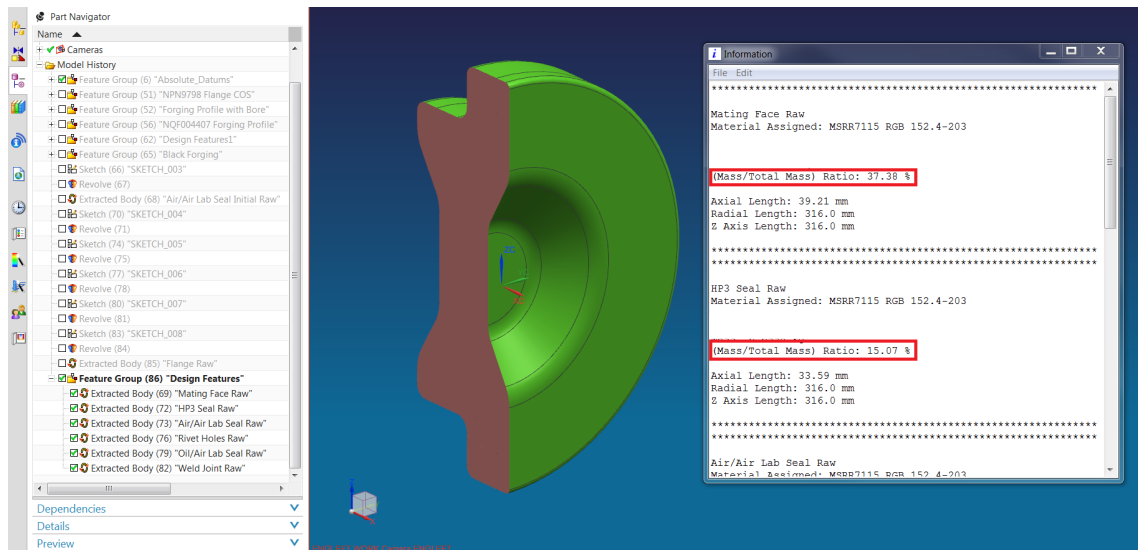


Figure 6.9: Condition of supply part feature geometry definition and the calculated representative features mass ratio.

6.2 IPT Stub Shaft Case Study

6.2.1 The Set-up

The aim of this case study is to demonstrate the methodology improvements and to showcase the scalability of the framework. The IPT Stub Shaft is not a critical component and is classed as a secondary component in Rolls-Royce's terms. However, the geometry and the manufacturing process is more complex than the Rear Lock Plate. The primary function of the stub shaft is to radially locate the Internal Pressure Turbine and transfer loads from the disc to the bearing. The IPT stub shaft is illustrated in Figure 6.10. This case study assumes that the design of the stub shaft has been frozen, the requirements have been captured, final DFMEA analysis has been completed and the cost models have been built.

6.2.2 The Case Study

During the IPT stub shaft development process a variety of tools have been used at different stages of the lifecycle. The tools used in the product development process generated data specific to one of the following domains: requirements domain, design domain, manufacturing domain and cost domain. The code, that captures the targeted data and structures it, has been embedded in each of the tools (e.g. DOORS, NX CAD and Vanguard Studio) that have been used to generate the domain specific data. The

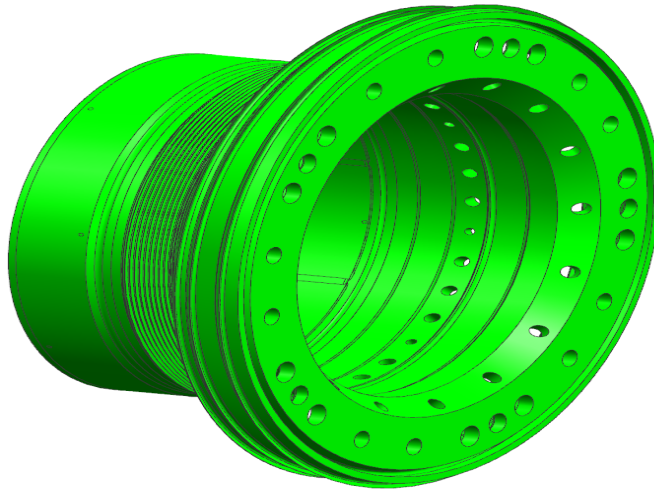


Figure 6.10: IPT Stub Shaft.

captured and structured data from all the domains has been saved into the CSV file format. This data has then been used to generate the map in BOXARR where links have been created between the requirements, design and cost domains. The following sections describe the processes within each of the domains.

6.2.2.1 Requirements Domain

Due to the fact that the IPT stub shaft is a secondary component, the requirements are often merged with a mini subsystem requirements or an interface requirements where few components are involved. In this study the IPT stub shaft requirements are merged with the Triple Seal Flange Joint requirements document. This study looks at the IPT stub shaft functional requirements only. Another study, presented in Section 6.3, investigates the Triple Seal Flange Joint interface and how it affects the calculation of the requirements impact when more components are added to the interdependency map.

6.2.2.2 Design Domain

In the design domain the modelled geometry always represents the finished component. However, in the manufacturing domain intermediate steps are necessary in order to produce the final shape of the component. Therefore, other geometries are needed to represent different component states at different manufacturing stages. In other words, the component's transformation journey from a lump of metal to a finished part has to be simulated. This approach is desired, because it helps to identify which manufacturing processes are best for a particular geometric shape and to understand the major cost drivers of these manufacturing processes. The IPT stub shaft is split into

two sub-components which are Flange component and Bearing Race component. The main reason for having two components is because two different materials are used to manufacture them. In addition, there are three intermediate geometries that represent different component states at different manufacturing stages. Figure 6.11 illustrates the intermediate geometries of the IPT stub shaft.

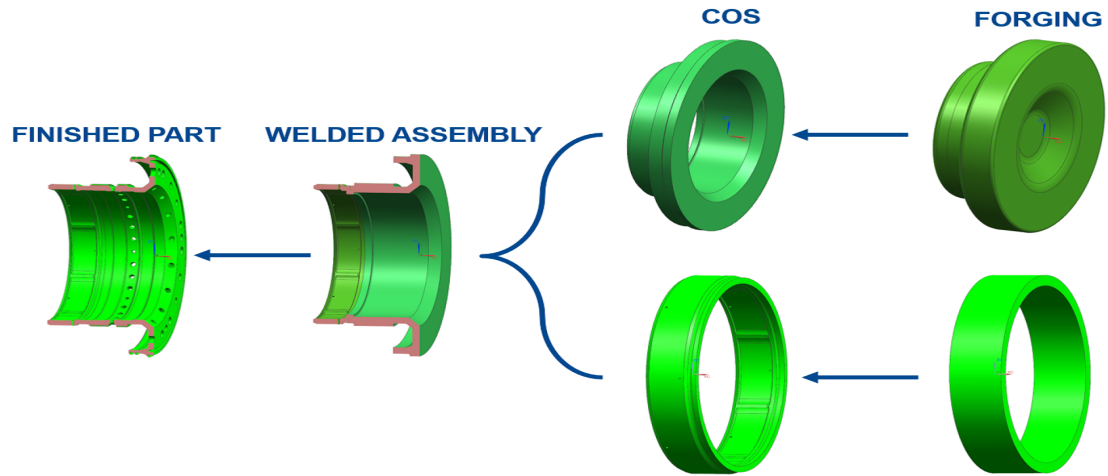


Figure 6.11: Intermediate geometries of the IPT Stub Shaft.

As seen from Figure 6.11 the Flange component has the Forging geometry which is produced by the closed die forging manufacturing process and the Bearing Race is a forged ring. Machining operations are used in order to bring the geometry to the condition of supply state. Then two sub-components are welded together to produce a single component. Further machining operations are necessary to produce the final geometry of the component.

The suggested improvement, to define the geometric features, has been applied to all of the models. The defined geometric features for the Forging models and the Finished component model are shown in Figures 6.12 and 6.13, respectively. The design features data has been extracted by the code (Appendix B) embedded in the NX CAD software and saved into CSV file format for all three geometry models.

In addition, the features mass ratio data that is used for features material cost calculation has been extracted from the forging models only.

6.2.2.3 Cost Domain

Vanguard Cost models have been created to simulate the manufacturing sequences and cost of the forging geometry, COS geometry and finished part geometry. The improvement that helped to speed up the feature tagging process has been implemented. As

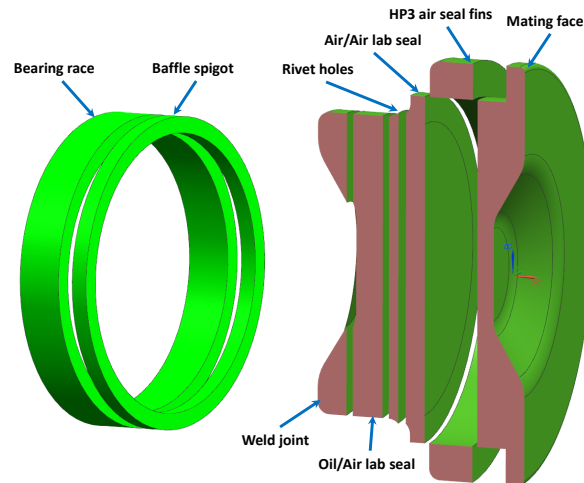


Figure 6.12: Geometric features of the IPT Stub Shaft forging model.

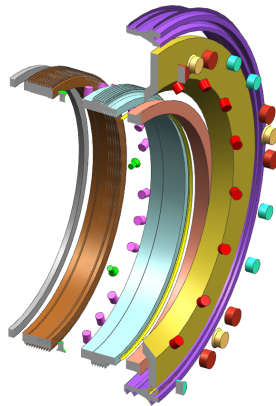


Figure 6.13: IPT Stub Shaft Design Features. Coloured names represent the stub shaft features - HP3 Air Seal Fins, Flange Holes, Taper Bolt Holes, Dowel Holes, R030 Holes, Mating Face, Seal Ring Mating Face, Air/Air Lab Seal, Rivet Holes, R044 Holes, Oil/Air Lab Seal, Weld Joint.

shown in Figure 6.14, five cost models have been built. The relevant data has been extracted and saved into CSV file format. Three types of data have been captured: data necessary to create value streams, manufacturing operation sequence data and cost data.

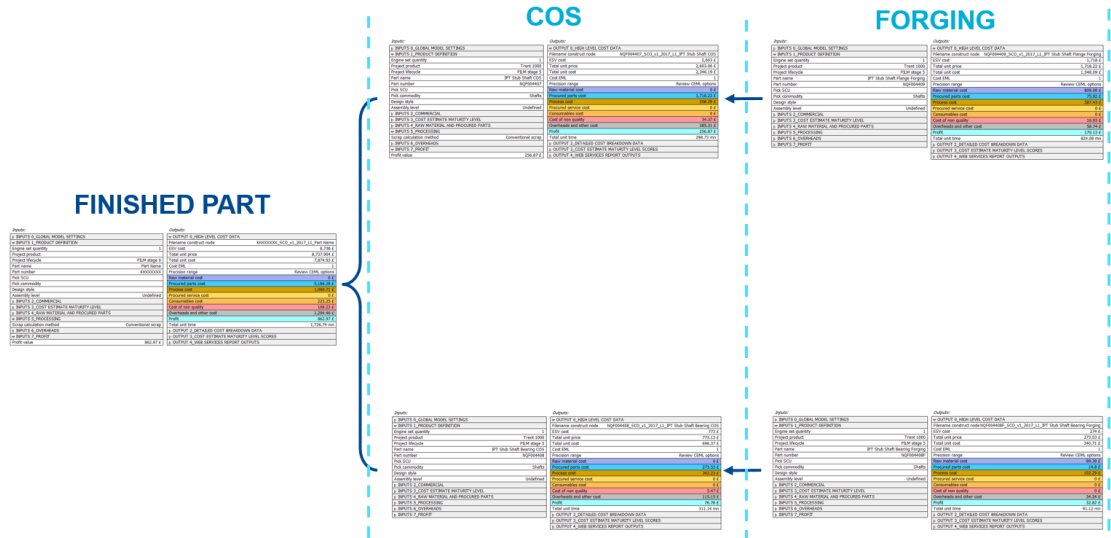


Figure 6.14: Vanguard cost models for different geometries. All costs are representative.

6.2.2.4 BOXARR

The CSV files with the specific structured data generated from the requirements domain, design domain and cost domain have been uploaded to the BOXARR. The file upload sequence is presented in Appendix G. Although, the sequence is not important from the BOXARR's point of view the proposed sequence frames the product development process. Following the upload sequence and referring back to the methodology framework in Figure 4.1, the scripts written in BOXARR automates the processes from stage 3 through to stage 7. As a result, the data structure for requirements domain and design domain has been created; the links between those domains have been added; the data structure for cost domain together with the links between them have been created. Once all the necessary files have been uploaded the requirements impact has been calculated. The top level view of the IPT stub shaft knowledge map is shown in Figure 6.15.

6.2.3 Result and Discussion

This case study demonstrated the improved knowledge capture methodology framework applied to the IPT stub shaft. The suggested improvements enabled full automation of the knowledge capture process and enhanced the cost data by assigning material cost to the relevant features. The benefit of knowing the magnitude of the feature material cost is that it allows the designer or manufacturing engineer quickly identify the potential areas on the geometry where the amount of material can be reduced. For example, during the cost reduction activity, the normal forging geometry is reduced to slimline

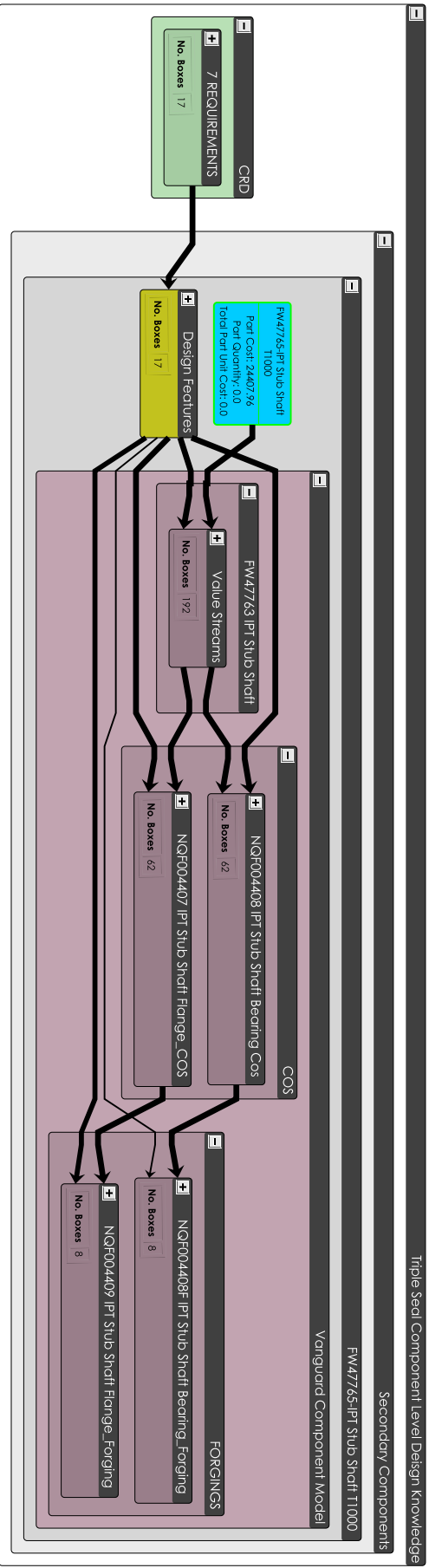


Figure 6.15: IPT Stub Shaft knowledge map with the links between requirements domain (green colour), design domain (yellow colour) and cost domain (dark pink colour). The costs are representative.

forging geometry by targeting the most costly features first. Another benefit is that the cost of the feature is captured not only at the finished part level but a total cost of the feature is captured from the different cost models. In other words, the cost is captured from all the value adding manufacturing operations that are required to produce the feature. As seen in Figure 6.15, the cost of the features is summed from five different cost models. The total cost of each feature and the material cost of the feature are shown in Figure 6.16. As illustrated in Figure 6.17, the amount of material that needs to be removed in order to produce the ‘*Mating Face*’ feature is the largest. Therefore, the associated manufacturing costs as well as the material cost are also the highest as shown in Figure 6.16. The comment received from the engineers at Rolls-Royce was, although it is intuitive, that for example, the ‘*Mating Face*’ feature has the highest cost it is not clear what that cost value is. This often leads to rough order of magnitude (ROM) estimates. ROM by definition will never be accurate. This is one of the reasons why the projects are missing the targets. It can be concluded, that by knowing the cost of each feature better decisions, as well as more accurate cost estimates, can be made. In the author’s opinion, knowing the requirements impact, as well as being able to trace the factors that impact the requirements, will improve the decision making process. For example, in early product development (i.e. redesign) stages the requirements impact will give an idea which requirements have the highest number of interdependencies (both from the requirements and features point of view). Moreover, it will highlight the requirements that affect the high cost features. Having this kind of information helps engineers in requirements prioritisation by focusing on high impact requirements first. Furthermore, being able to quickly trace the links to other requirements (through design features) or to features themselves should reduce the overall product development time considerably.

In this research, the factors that impact the requirements are the requirements importance (defined by modal words), requirements complexity (how many features are linked to one requirement), features complexity (how many requirements are linked to one feature) and the cost of the feature(s) that the requirement is linked to. The impact of the IPT stub shaft functional requirements is illustrated in Figure 6.18. As seen in the figure, the requirement *EDNS01000309547-039 (Shall seal air between the Upstream Cavity and Stub Shaft Cavity)* (further referred as requirement 39) has the highest impact of 12.70% and the requirement *EDNS01000309547-037 (Shall transfer air across the downstream cavity (vortex loss))* (further referred as requirement 37) comes second with the impact of 11.73%. Figure 6.19 shows the filtered requirements in BOXARR. As seen in the figure, the requirement 39 has links to two features (‘*Mating Face*’ and ‘*HP3 Air Seal Fins*’); and the requirement 37 has links to three features (‘*HP3 Air Seal Fins*’, ‘*Air/Air Lab Seal*’ and ‘*R030 Holes*’). The combined features cost of the requirement 39 is higher than the cost of the requirement 37 features. Therefore, the overall *Weight on Features Cost* is higher for the requirement 39. The overall *Weight on Requirements Complexity*, however, is higher for the requirement 37, because there are three links to features compared to two links for the requirement 39. The *Weight on*

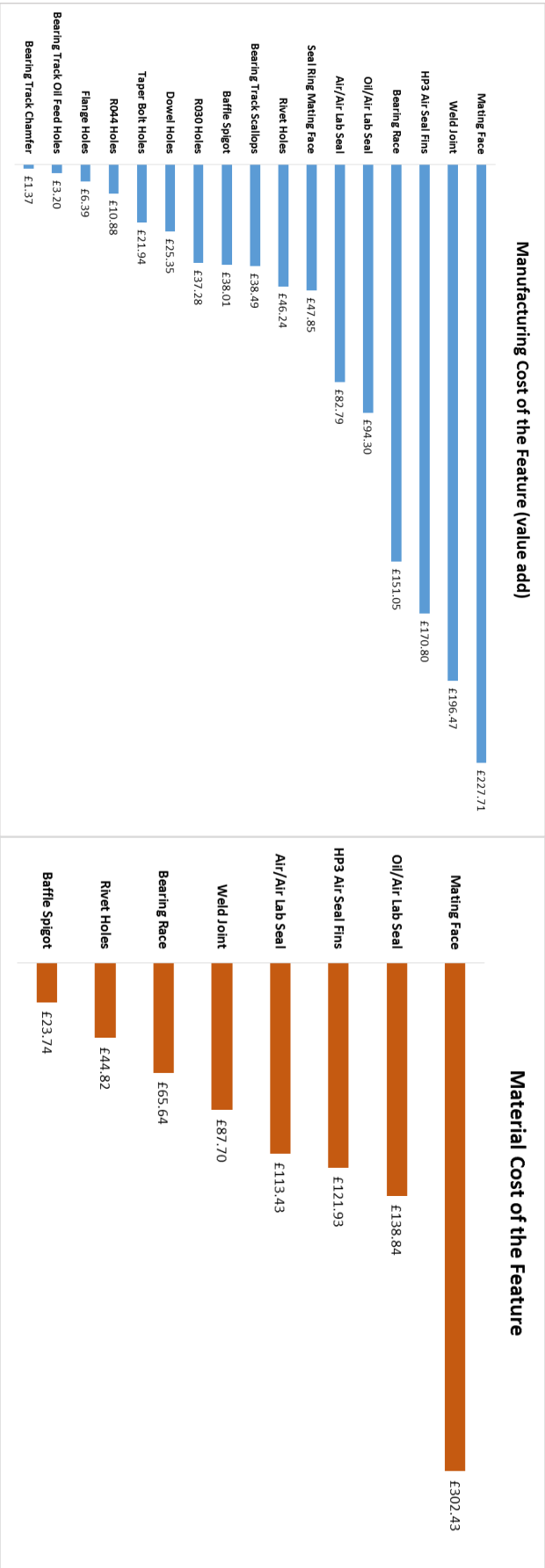


Figure 6.16: Representative features manufacturing and material cost of the IPT Stub Shaft.

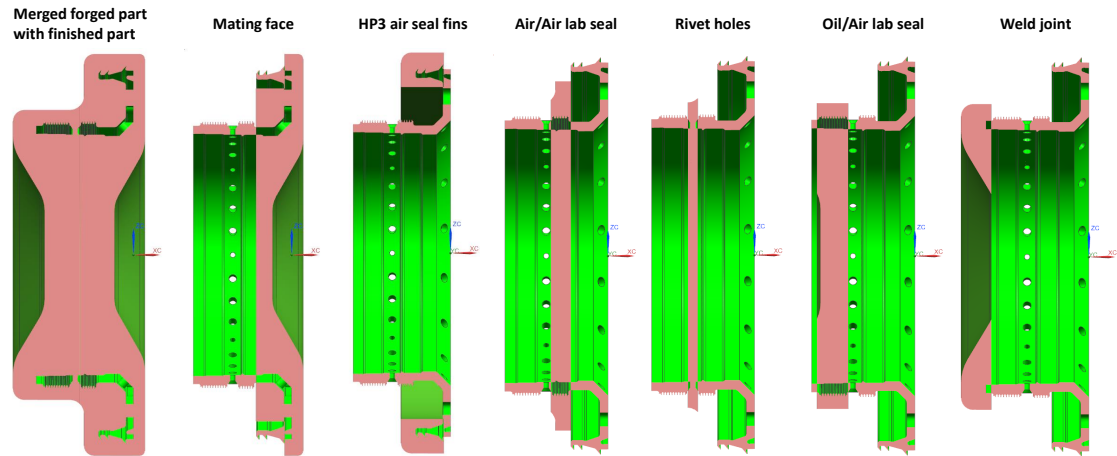


Figure 6.17: The amount of material that needs to be removed for each feature in order to get from the IPT Stub Shaft forging geometry to finished part geometry.

Features Complexity is equal for both, the requirement 39 and 37. Because the features of the requirements 39 and 37 have a total of six links to other features. Figure 6.20 illustrates filtered links of the ‘HP3 Air Seal Fins’ feature in BOXARR. As seen in the figure, there are links to three different requirements. Similarly, feature ‘Air/Air Lab Seal’ has two links and feature ‘R030 Holes’ has one link to the requirements. For the requirement 39, feature ‘Mating Face’ has three links and feature ‘HP3 Air Seal Fins’ has three links to the requirements.

Weight on Features Complexity is important, because it highlights the number of interdependencies between the requirements. The higher the weight factor, the more interdependencies there are. The *Requirements Importance Weight* is 9 for both, requirement 39 and 37. Because these are the functional requirements, they carry the highest weight, as described in Section 4.7. All the parameters that are needed to calculate the impact of each functional requirement for the IPT stub shaft, using Equation 4.11, are summarised in Figure 6.21.

BOXARR filtering and visualisation capabilities improve the navigation efficiency. It means that the time that otherwise had been spent on data search activities can be used to do the value adding work.

Product development process, especially in the aerospace industry, can take a number of years to complete. During that process large amounts of data are generated. Therefore, having a good knowledge base at the start of a new project is essential for the project to be successful. This case study has demonstrated that it is possible to capture the design knowledge without any significant interference in the product development work flow. In other words, the proposed methodology does not prescribe the specific method that needs to be followed in order to capture the design knowledge. In fact, it is a descriptive approach where the specific knowledge is captured from product across the

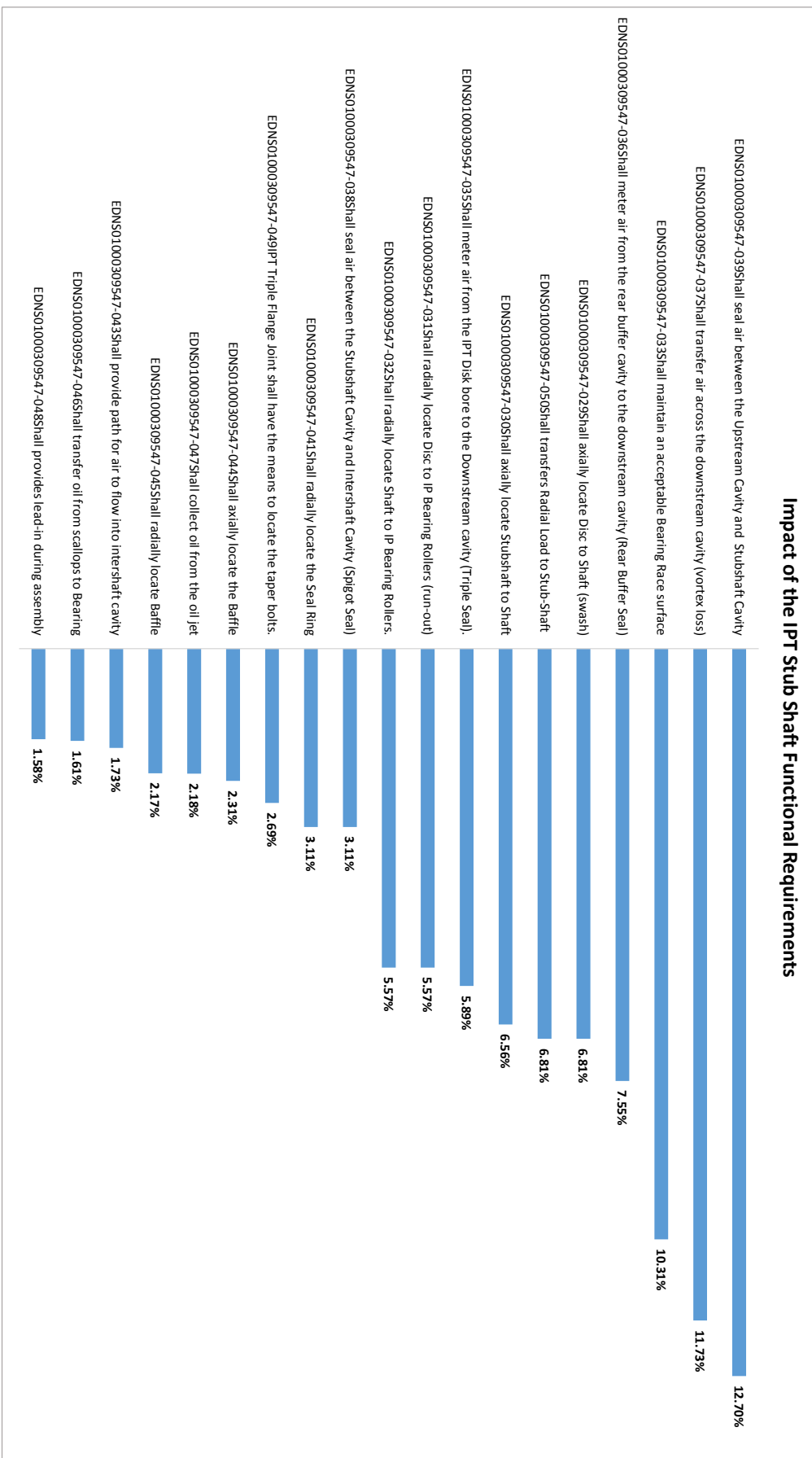
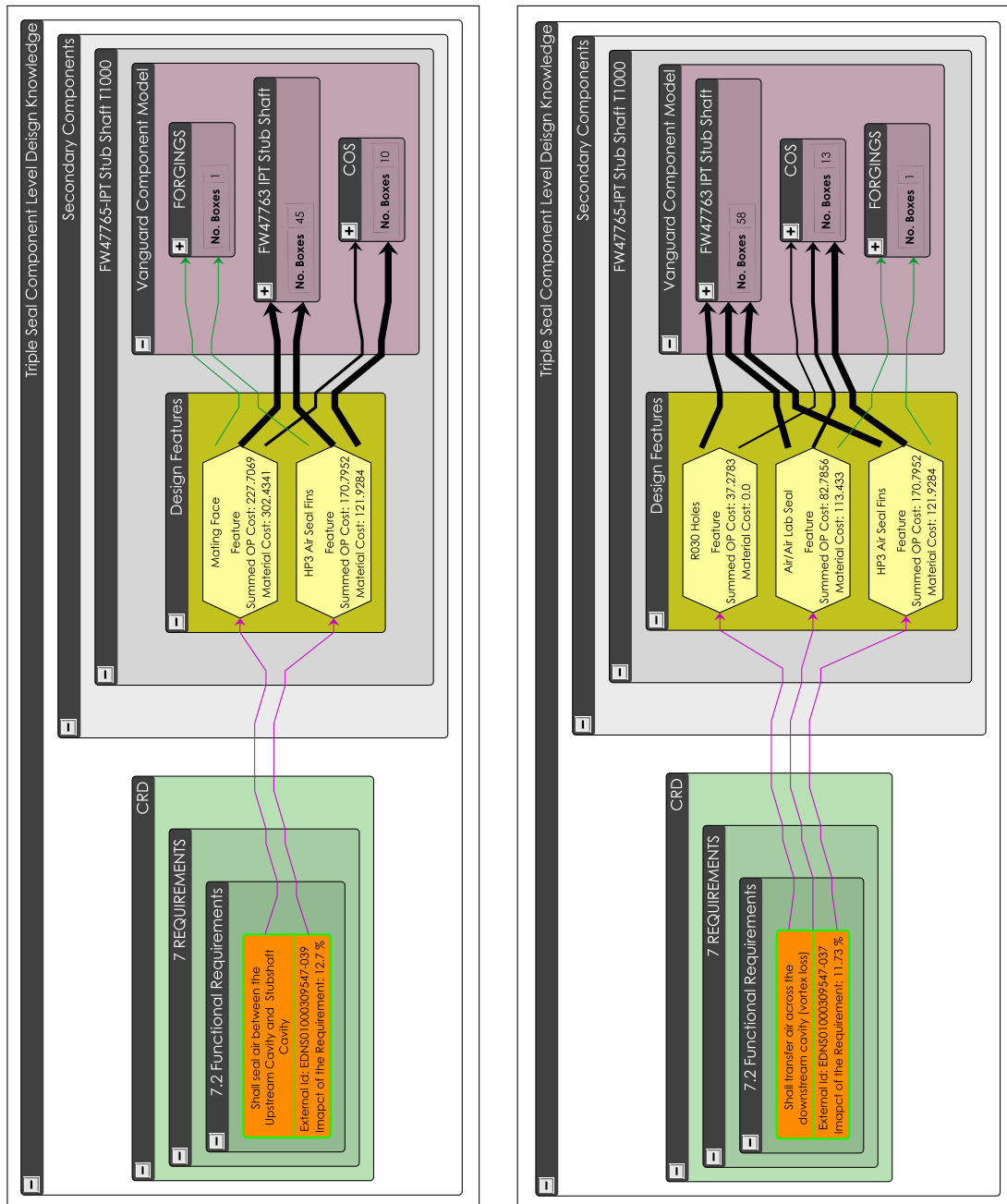


Figure 6.18: The impact of the IPT Stub Shaft functional requirements.

Figure 6.19: Filtered requirements (*EDNS01000309547-039* and *EDNS01000309547-037*) links to features.

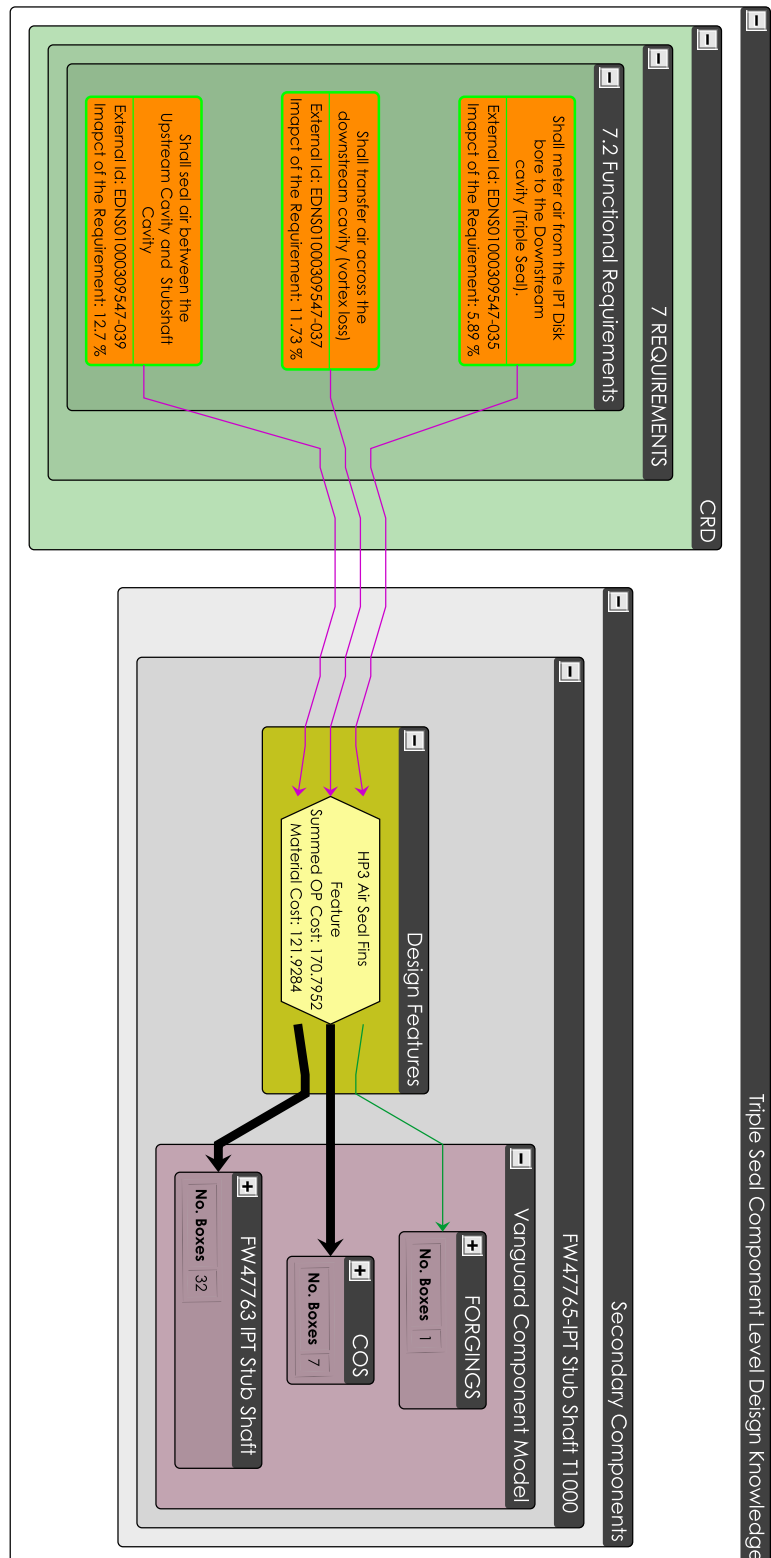


Figure 6.20: The number of requirements that are linked to the 'HP3 Air Seal Fins' feature.

Requirements ID	Weights on Features Complexity (F_c)	Weights on Features Cost (FC_w)	Number of Links to Features	Weights on Requirements Complexity (R_c)	Requirements Importance Weight (R_i)	Impact
EDNS01000309547-039	2.16	2.99	2	0.72	9	12.70%
EDNS01000309547-037	2.16	2.18	3	1.08	9	11.73%
EDNS01000309547-033	1.44	2.61	2	0.72	9	10.31%
EDNS01000309547-036	1.44	1.33	2	0.72	9	7.55%
EDNS01000309547-029	1.08	1.71	1	0.36	9	6.81%
EDNS01000309547-050	1.08	1.71	1	0.36	9	6.81%
EDNS01000309547-030	1.44	0.87	2	0.72	9	6.56%
EDNS01000309547-035	1.08	1.28	1	0.36	9	5.89%
EDNS01000309547-031	1.08	1.13	1	0.36	9	5.57%
EDNS01000309547-032	1.08	1.13	1	0.36	9	5.57%
EDNS01000309547-038	0.72	0.36	1	0.36	9	3.11%
EDNS01000309547-041	0.72	0.36	1	0.36	9	3.11%
EDNS01000309547-049	0.72	0.16	1	0.36	9	2.69%
EDNS01000309547-044	0.36	0.35	1	0.36	9	2.31%
EDNS01000309547-047	0.36	0.29	1	0.36	9	2.18%
EDNS01000309547-045	0.36	0.29	1	0.36	9	2.17%
EDNS01000309547-043	0.36	0.08	1	0.36	9	1.73%
EDNS01000309547-046	0.36	0.02	1	0.36	9	1.61%
EDNS01000309547-048	0.36	0.01	1	0.36	9	1.58%

Figure 6.21: The summary of the parameters needed to calculate the impact of each functional requirement for the IPT stub shaft.

requirements, design and cost domains, structured and reproduced as an interdependency map between those domains. This case study also proved that no extra work is required for more geometrically complex components and it is scalable on any number of requirements, features or cost models. This case study has also demonstrated that requirements impact can highlight the requirements that are linked to the high cost features as well as highlight the number of interdependencies between the requirements. The following case study investigates how the requirements impact changes when the proposed methodology is applied to a sub-system of components. The IPT stub shaft is a part of the interface sub-system in the new study.

6.3 Triple Seal Flange Joint Case Study

6.3.1 The Case Study

The triple seal flange joint solution introduces a triple taper bolted IP Turbine flange pack which sees the Stub-shaft, Disc and Shaft all radially located through the taper bolts. This removes the thermal mismatch effect and should prevent any slippage within the flange pack. Secondary air sealing is maintained through inclusion of a new ring seal. Figure 6.22 shows the system boundary of the triple seal flange joint interface. As seen in the figure, there are three main components in this system: stub shaft, shaft and

disc. The taper bolt is not taken into account in this study, because its effect on the requirements impact is negligible.

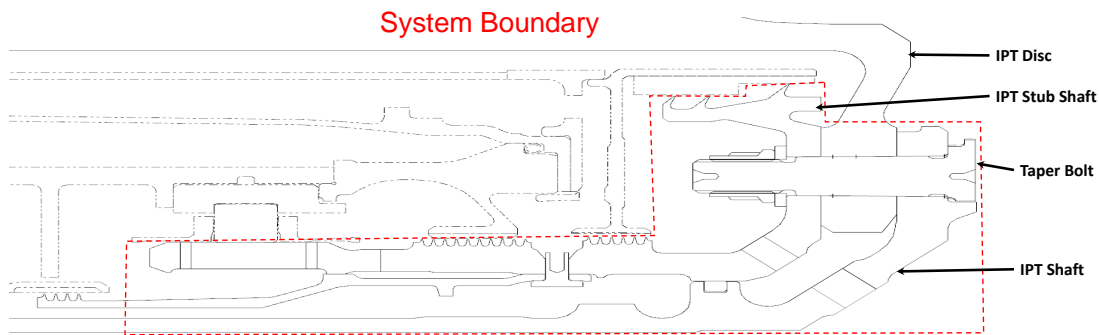


Figure 6.22: System boundary of the triple seal flange joint interface.

The main aim of this case study is to investigate the requirements impact of the triple seal flange joint interface and to compare with the case study results of the IPT stub shaft. In the latter case study the triple seal flange joint interface requirements document was used. Therefore, one of the objectives of this study is to understand if the change in the requirements impact reflects the effects of the additional components (shaft and disc).

The improved methodology has been used to capture the relevant data for the IPT stub shaft, shaft and disc design process. The data has then been imported into BOXARR to create the interdependency map, between the requirements, design and cost domains, that represents the triple seal flange joint system. The new design knowledge map is illustrated in Figure 6.23. The cost values are modified due to Rolls-Royce intellectual property rights. As seen in Figure 6.23, the design knowledge of the sub-system is presented in an elegant and concise manner. Having the requirements, design and cost data on one screen and the capability to easily navigate through the data saves tremendous amounts of time. The evidence from the academia and from industry, mentioned in this research, proved the fact that to find the complete set of information about the design process is difficult. The information is often fragmented and comes from the different data sources. All the pieces have to be put together in order to make sense of the gathered information.

The following section investigates the impact of the requirements and compares with the IPT stub shaft case study results.

6.3.2 Result and Discussion

The aim of the triple seal flange joint case study has been to investigate how the requirements impact changes going from a single component system to a system of components. Both case studies have used the same requirements document (Triple Seal Flange Joint

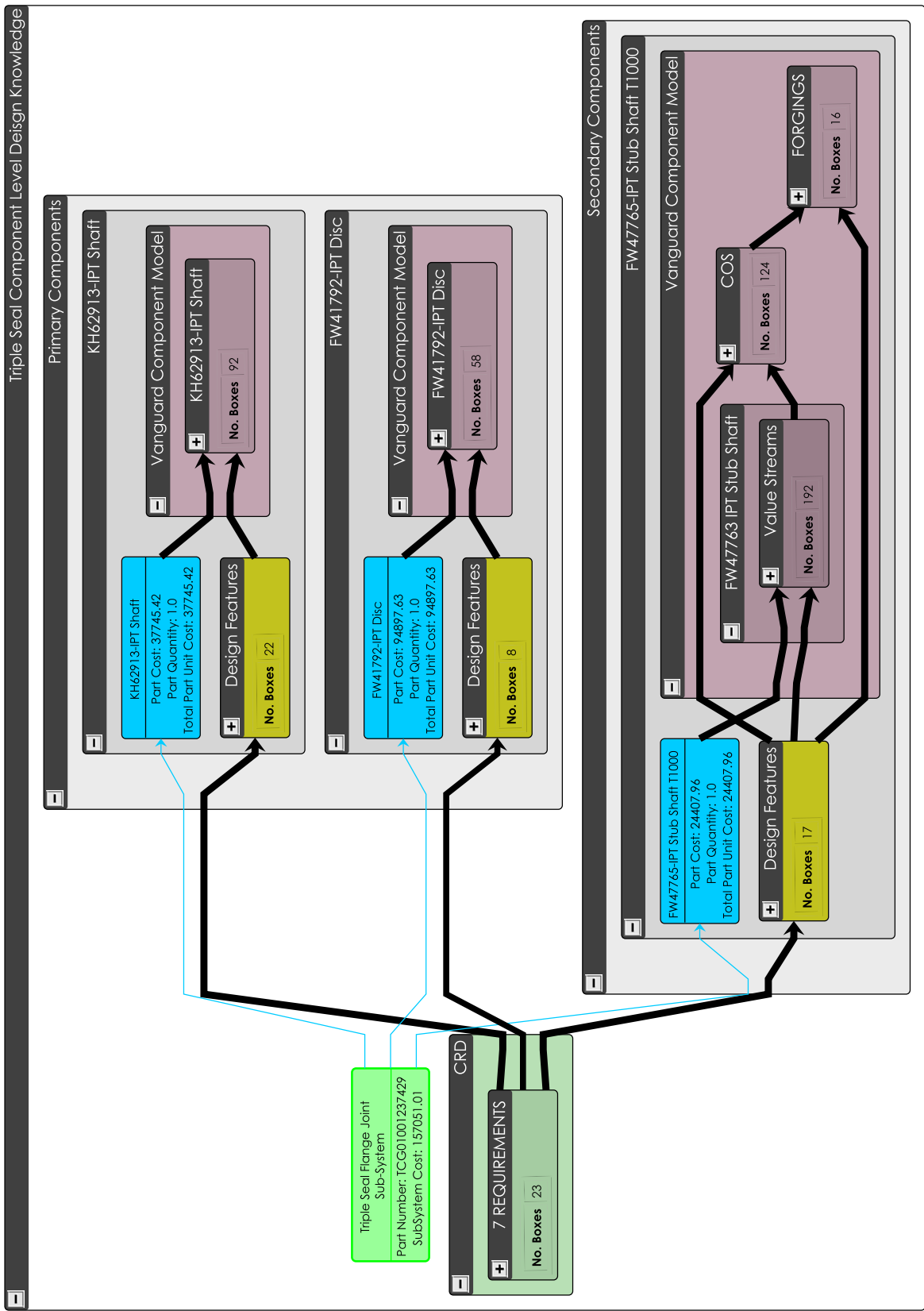


Figure 6.23: Design knowledge of the triple seal flange joint interface sub-system. The costs shown are examples and not the real values.

Requirements Document). The summary of requirements impact and the weights for each requirement are presented in Figures 6.24 and 6.25, respectively.

The feature costs of the stub shaft, shaft and disc are shown in Figure 6.26. The red bars in the figure highlight the features that are relevant to this case study.

The top five requirements from Figure 6.25 have been investigated. The requirements are (as listed in Figure 6.25):

- EDNS01000309547-039 *Shall seal air between the Upstream Cavity and Stubshaft Cavity.*
- EDNS01000309547-029 *Shall axially locate Disc to Shaft (swash).*
- EDNS01000309547-034 *Shall transfer torque from Disk to Shaft.*
- EDNS01000309547-050 *Shall transfers Radial Load to Stub-Shaft.*
- EDNS01000309547-037 *Shall transfer air across the downstream cavity (vortex loss).*

Summarised results for analysis are presented in Figure 6.27. As seen in the figure, requirements 39, 29 and 34 each have links to three features; requirement 50 has links to two features; and requirement 37 has links to four features. There are a total of 39 links from the requirements to features. Normalising the number of links from each requirement against the total number of links *Weight on Requirements Complexity* (R_C) has been calculated. This parameter highlights the number of features that are affected by a single requirement. The R_C for the requirements 39, 29 and 34 is 0.69; 0.46 for requirement 50; and 0.92 for requirement 37. However, each feature is affected by different requirements.

As presented in Figure 6.28, requirement 39 has links to three features, but each feature is linked to different requirements. For example, in addition to the requirement 39 the feature ‘*Mating Face*’ also has links to the requirements 29 and 50. Similarly, requirements 35, 37 and 39 are linked to the feature ‘*HP3 Air Seal Fins*’; and five requirements (e.g. 29, 34, 39, 40 and 50) are linked to the feature ‘*Flange*’. There is a total of 11 links. However, some of these links are duplicates. As shown in Figure 6.28, requirement 39 is interdependent with six other requirements and there are five duplicate links. This interdependency is measured by the ‘*Weight on Features Complexity*’ (F_C). To calculate the F_C the number of links from the requirements to a feature is normalised against the total number of links from the requirements to the features. Hence, as shown in Figure 6.27, the F_C of the requirement 39 is 2.54; requirement 29 is 2.54; requirement 34 is 2.08; requirement 50 is 1.85; and requirement 37 is 1.62. The F_C is an important parameter, it considers the total number of relational links as well as the number of true interdependencies. Therefore, the higher the F_C value the more interdependencies there are between the requirements.

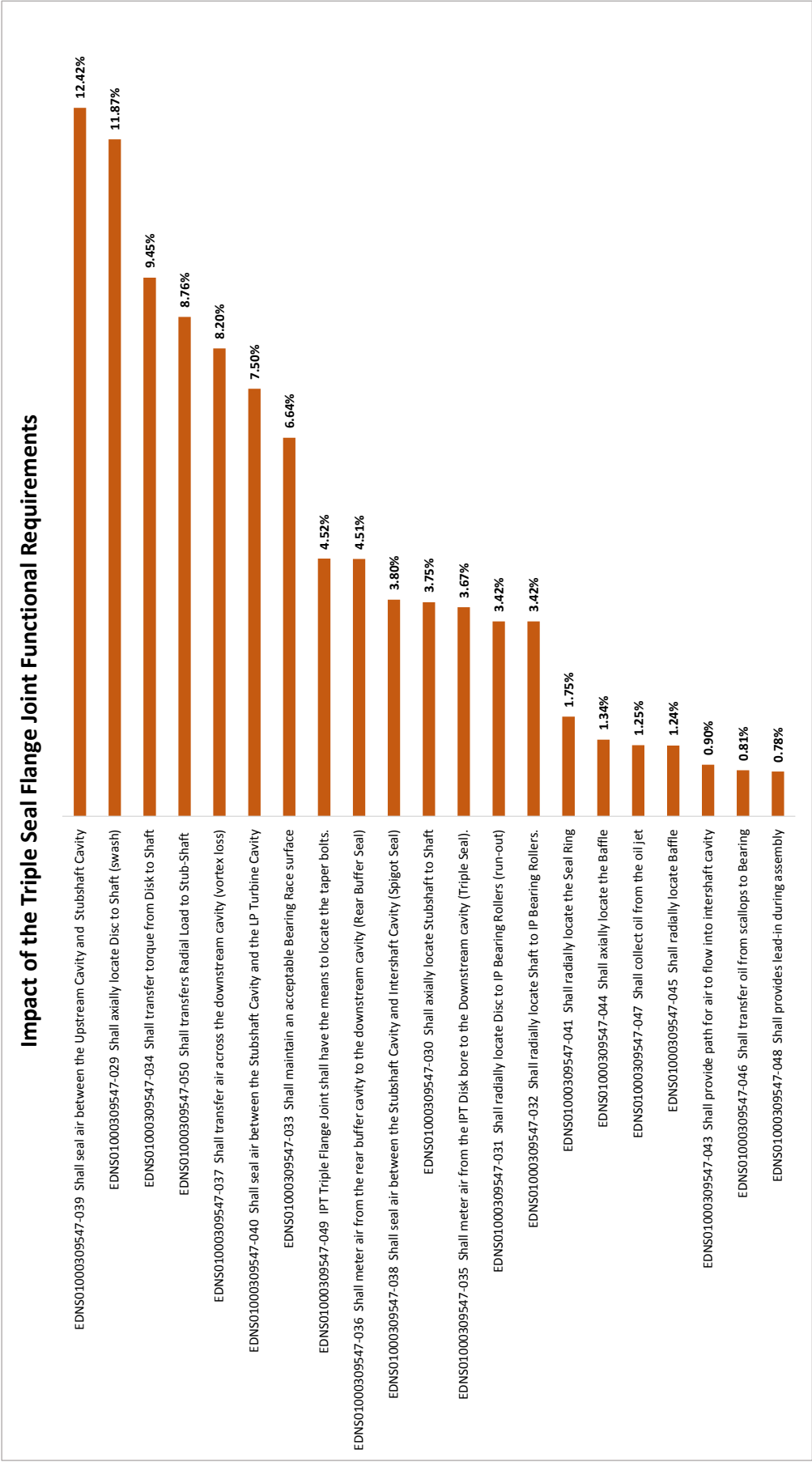


Figure 6.24: The impact of the triple seal flange joint functional requirements.

Requirements ID	Weights on Features Complexity (F_C)	Weights on Features Cost (FC_W)	Number of Links to Features	Weights on Requirements Complexity (R_C)	Requirements Importance Weight (R_I)	Impact
EDNS01000309547-039	2.54	4.24	3	0.69	9	12.42%
EDNS01000309547-029	2.54	3.91	3	0.69	9	11.87%
EDNS01000309547-034	2.08	2.91	3	0.69	9	9.45%
EDNS01000309547-050	1.85	2.96	2	0.46	9	8.76%
EDNS01000309547-037	1.62	2.39	4	0.92	9	8.20%
EDNS01000309547-040	1.85	2.20	2	0.46	9	7.50%
EDNS01000309547-033	0.92	2.61	2	0.46	9	6.64%
EDNS01000309547-049	0.92	1.10	3	0.69	9	4.52%
EDNS01000309547-036	0.92	1.33	2	0.46	9	4.51%
EDNS01000309547-038	0.92	0.67	3	0.69	9	3.80%
EDNS01000309547-030	0.92	0.87	2	0.46	9	3.75%
EDNS01000309547-035	0.69	1.28	1	0.23	9	3.67%
EDNS01000309547-031	0.69	1.13	1	0.23	9	3.42%
EDNS01000309547-032	0.69	1.13	1	0.23	9	3.42%
EDNS01000309547-041	0.46	0.36	1	0.23	9	1.75%
EDNS01000309547-044	0.23	0.35	1	0.23	9	1.34%
EDNS01000309547-047	0.23	0.29	1	0.23	9	1.25%
EDNS01000309547-045	0.23	0.29	1	0.23	9	1.24%
EDNS01000309547-043	0.23	0.08	1	0.23	9	0.90%
EDNS01000309547-046	0.23	0.02	1	0.23	9	0.81%
EDNS01000309547-048	0.23	0.01	1	0.23	9	0.78%

Figure 6.25: The summary of the parameters needed to calculate the impact of each functional requirement for the triple seal flange joint.

The ‘*Weight on Features Cost*’ (FC_W) captures the cost aspect of each feature. As shown in Figure 6.27, requirement 39 is linked to three features. Each feature has a manufacturing cost assigned to them. The cost of the individual feature is normalised against the total ‘value add’ processing cost of a particular component. Hence, the FC_W of the requirement 39 is 4.24; requirement 29 is 3.91; requirement 34 is 2.91; requirement 50 is 2.96; and requirement 37 is 2.39.

The proposed methodology considers *Functional Requirements* only. Therefore, the *Requirements Importance Weight* (R_I), as defined in Section 4.7 is 9 for all the functional requirements.

The impact of each requirement has been calculated using the Equation 4.11. Requirements 39 and 29 have the highest impact of 12.42% and 11.87%, respectively. As shown in Figure 6.27, the main differentiators are high features cost and high requirements interdependency. Comparing the parameters of the requirements 34 and 50, it is clear that requirement 34 has a much higher interdependency and requirement 50 is linked to the higher cost features. However, more links to the features and high interdependency of the requirement 34 equates to a bigger impact than the high features cost of the requirement 50. Therefore, requirement 34 is the third high impact requirement with the value of 9.45% and requirement 50 is the fourth, with the value of 8.76%. On the other hand, requirement 37 is linked to the highest number of features, but the features cost

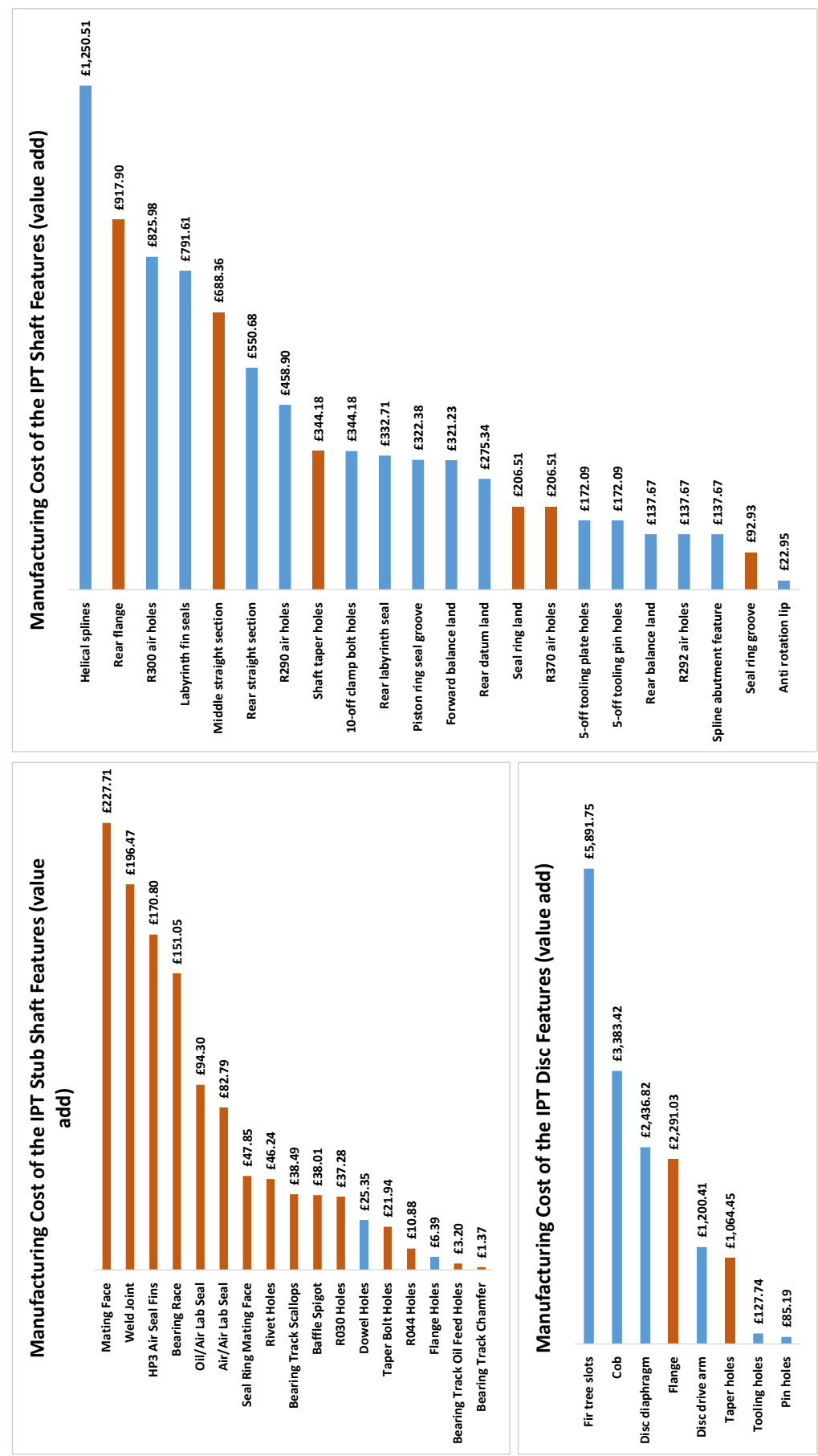


Figure 6.26: The summary of the representative stub shaft, shaft and disc feature costs.

Requirements ID	Number of Links to Features	Features (component it belongs to)	Number of Links to Requirements (from features)	Weights on Features Complexity (F_c)	Normalised Feature Costs	Weights on Features Cost (F_{C_w})	Weights on Requirements Complexity (R_c)	Impact
EDNS01000309547-039	3	Mating Face (Stub Shaft)	3	2.54	0.19	4.24	0.69	12.42%
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		Flange (Disc)	5		0.14			
		Mating Face (Stub Shaft)	3		0.19			
EDNS01000309547-029	3	Rear Flange (Shaft)	3	2.54	0.11	3.91	0.69	11.87%
		Flange (Disc)	5		0.14			
		Middle Staright Section (Shaft)	1		0.08			
		Rear Flange (Shaft)	3		0.11			
EDNS01000309547-034	3	Flange (Disc)	5	2.08	0.14	2.91	0.69	9.45%
		Mating Face (Stub Shaft)	3		0.19			
		Flange (Disc)	5		0.14			
		Mating Face (Stub Shaft)	3		0.19			
EDNS01000309547-050	2	Flange (Disc)	5	1.85	0.14	2.96	0.46	8.76%
		R370 Air Holes (Shaft)	1		0.02			
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		Air/Air Lab Seal (Stub Shaft)	2		0.07			
EDNS01000309547-037	4	R030 Holes (Stub Shaft)	1	1.62	0.03	2.39	0.92	8.20%
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		Air/Air Lab Seal (Stub Shaft)	2		0.07			
		R030 Holes (Stub Shaft)	1		0.03			

Figure 6.27: The summary of the parameters for five requirements used to calculate the requirements impact.

Requirements ID	Features (component it belongs to)	Number of Links to Requirements (from features)	Requirements Linked to a Feature	Total Number of Links from Requirements to features	Number of True Interdependencies	Number of Duplicate Interdependencies	Interdependent Requirements
EDNS01000309547-039	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050	11	6	5	EDNS01000309547-029 EDNS01000309547-050 EDNS01000309547-035 EDNS01000309547-037 EDNS01000309547-034 EDNS01000309547-040
	HP3 Air Seal Fins (Stub Shaft)	3	EDNS01000309547-035 EDNS01000309547-037 EDNS01000309547-039				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
	Rear Flange (Shaft)	3	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-040				
EDNS01000309547-029	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050	11	4	7	EDNS01000309547-039 EDNS01000309547-050 EDNS01000309547-034 EDNS01000309547-040
	Middle Straight Section (Shaft)	1	EDNS01000309547-034				
	Rear Flange (Shaft)	3	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-040				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
EDNS01000309547-034	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050	9	4	5	EDNS01000309547-029 EDNS01000309547-040 EDNS01000309547-039 EDNS01000309547-050
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
EDNS01000309547-050	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050	8	4	4	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-034 EDNS01000309547-040
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
	Mating Face (Stub Shaft)	3	EDNS01000309547-029 EDNS01000309547-039 EDNS01000309547-050				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				
EDNS01000309547-037	R370 Air Holes (Shaft)	1	EDNS01000309547-037	7	3	4	EDNS01000309547-035 EDNS01000309547-039 EDNS01000309547-036 EDNS01000309547-037
	HP3 Air Seal Fins (Stub Shaft)	3	EDNS01000309547-035 EDNS01000309547-037 EDNS01000309547-039				
	Air/Air Lab Seal (Stub Shaft)	2	EDNS01000309547-036 EDNS01000309547-037				
	R030 Holes (Stub Shaft)	1	EDNS01000309547-037				
	Flange (Disc)	5	EDNS01000309547-029 EDNS01000309547-034 EDNS01000309547-039 EDNS01000309547-040 EDNS01000309547-050				

Figure 6.28: Requirements interdependencies through features.

and requirements interdependency is low compared to other four requirements. Hence, it is the fifth high impact requirement with the value of 8.20%.

The following paragraph compares how requirements impact changes when more components are added to the system (i.e. going from a single component (stub shaft) to a multiple component system (stub shaft, shaft and disc)).

The parameters summary of the top five high impact requirements for both, the stub shaft and the triple seal flange joint systems are shown in Figure 6.29. As seen in the figure, the same three requirements (e.g. 39, 29 and 37) remained in the top five high impact requirements list. Also two new requirements (e.g. 34 and 50), in the triple seal flange joint system, made their way into the top five requirements list, pushing out requirements 33 and 36. Requirement 39 remained the highest impact requirement. The link to the additional disc ‘*Flange*’ feature increased the number of interdependencies and the FC_W parameter reinforcing the impact of requirement 39. As seen in Figure 6.26, the value add cost of the disc ‘*Flange*’ feature is the highest in the triple seal flange joint system. Hence, a significant change in the FC_W value. It is important to note that the additional link from the requirement 39 to the disc ‘*Flange*’ feature has been captured and its effect propagated through the system yielding the highest requirement impact value as a result.

An even stronger effect of the additional links from requirement 29 to the shaft ‘*Rear Flange*’ feature and the disc ‘*Flange*’ feature can be observed in Figure 6.29. Both features are high cost features. The increase in cost has been captured by the FC_W parameter which increased from 1.71 (in the stub shaft system) to 3.91 (in the triple seal flange joint system). Furthermore, more links to the features have also increased the number of interdependencies leading to an increase of the FC parameter from 1.08 (in the stub shaft system) to 2.54 (in the triple seal flange joint system). These changes have been captured and translated into the higher impact value of requirement 29.

The impact of requirement 37, however, decreased from the second highest (in the stub shaft system) to the fifth highest (in the triple seal flange joint system). As seen in Figure 6.29, the number of links to the features increased in the triple seal flange joint system leading to an increase in the FC_W parameter. However, the total number of links from the requirements to the features have also increased from 25 (in the stub shaft system) to 39 (in the triple seal flange joint system). This is an important observation. The increased number of links defines the boundary of the system and allows the system to recalibrate itself from the single component to the multiple component system. Therefore, the interdependency (FC) value reduced relative to the triple seal flange joint system, as a result, from 2.16 to 1.62. This suggests, that there are higher interdependency requirements in the triple seal flange joint system than requirement 37. Due to the same reason the R_C parameter of the requirement 37 has also reduced from 1.08 (in the stub shaft system) to 0.92 (in the triple seal flange joint system).

This case study demonstrated how the value of the requirements impact changes when the same set of requirements is applied to a single component system (e.g. stub shaft)

Requirements ID	Number of Links to Features	Features (component it belongs to)	Number of Links to Requirements (from features)	Weights on Features Complexity (F_c)	Normalised Feature Costs	Weights on Features Cost (FC_w)	Weights on Requirements Complexity (R_c)	Impact
EDNS01000309547-039	2	Mating Face (Stub Shaft)	3	2.16	0.19	2.99	0.72	12.70%
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
EDNS01000309547-037	3	Air/Air Lab Seal (Stub Shaft)	2	2.16	0.07	2.18	1.08	11.73%
		R030 Holes (Stub Shaft)	1		0.03			
		Bearing Race (Stub Shaft)	3		0.13			
EDNS01000309547-033	2	Weld Joint (Stub Shaft)	1	1.44	0.16	2.61	0.72	10.31%
EDNS01000309547-036	2	Air/Air Lab Seal (Stub Shaft)	2	1.44	0.07	1.33	0.72	7.55%
		Oil/Air Lab Seal (Stub Shaft)	2		0.08			
EDNS01000309547-029	1	Mating Face (Stub Shaft)	1	1.08	0.19	1.71	0.36	6.81%

Requirements ID	Number of Links to Features	Features (component it belongs to)	Number of Links to Requirements (from features)	Weights on Features Complexity (F_c)	Normalised Feature Costs	Weights on Features Cost (FC_w)	Weights on Requirements Complexity (R_c)	Impact
EDNS01000309547-039	3	Mating Face (Stub Shaft)	3	2.54	0.19	4.24	0.69	12.42%
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		Flange (Disc)	5		0.14			
EDNS01000309547-029	3	Mating Face (Stub Shaft)	3	2.54	0.19	3.91	0.69	11.87%
		Rear Flange (Shaft)	3		0.11			
		Flange (Disc)	5		0.14			
EDNS01000309547-034	3	Middle Straight Section (Shaft)	1	2.08	0.08	2.91	0.69	9.45%
		Rear Flange (Shaft)	3		0.11			
		Flange (Disc)	5		0.14			
EDNS01000309547-050	2	Mating Face (Stub Shaft)	3	1.85	0.19	2.96	0.46	8.76%
		Flange (Disc)	5		0.14			
EDNS01000309547-037	4	R370 Air Holes (Shaft)	1	1.62	0.02	2.39	0.92	8.20%
		HP3 Air Seal Fins (Stub Shaft)	3		0.14			
		Air/Air Lab Seal (Stub Shaft)	2		0.07			
		R030 Holes (Stub Shaft)	1		0.03			

Figure 6.29: The summary of the parameters for top five high impact requirements. Stub shaft (top table) and triple seal flange joint system (bottom table).

and to a multiple component system (in the triple seal flange joint system). The change takes into account the cost of the features, the number of features that the requirement is linked to and the interdependency between the requirements. In addition, the transition aspect from a single system to a sub-system is also considered when calculating the impact of the requirement.

To summarise, the proposed methodology captures the design story of the triple seal flange joint in a concise manner, as shown in Figure 6.23. It also demonstrates that it is possible to capture the knowledge from a *product*. This design knowledge map answers four fundamental questions in the product development process: ‘*what*’, ‘*why*’, ‘*how*’ and ‘*how much*’. The requirements tell ‘*what*’ the system does; the design features are there to satisfy the requirements (answer the ‘*why*’ question); the manufacturing sequence describes ‘*how*’ the features are manufactured; and the cost to manufacture the component, a sub-system or a system answers the ‘*how much*’ question.

BOXARR’s capability to tackle complexity provided real benefits when applying the methodology to the three case studies. This study, however, only looked at a three component sub-system, but applying this methodology to a much bigger sub-system or a system should offer the greatest benefit. Moreover, having the design knowledge map of the system provides instant access to the requirements, design features and cost data, enables engineers to investigate the interdependency links, and helps to understand the impact of the requirements.

Chapter 7

Conclusions and Future Work

In this chapter the conclusions of this thesis are presented by answering the research questions. The chapter ends with the recommendations for future research which could build on the findings of this work.

7.1 Conclusions

Over the last three decades researchers have proposed and developed a number of tools and methodologies to capture, retrieve and represent the design rationale throughout the design process. However, only a few successful applications of the design rationale system have made it into practical use in industry. While the majority of the developed tools and methodologies were aimed at capturing and representing design rationale and knowledge from people and processes, only a few researchers proposed to link the design knowledge directly to the product model. Most design projects are dominated by the redesign activities and only a few are aimed at developing totally new products. The information about the product, such as function, purpose, manufacturing route and cost is extremely important in the early stages of the new product development or product redesign process. Investigations by other researchers revealed that the available information sources did not support an effective re-use of the design knowledge and information. Moreover, they identified that only fragments of the design process were captured. While a number of methods to capture and represent design knowledge and methods to aid the decision making were proposed by the research community, all of them focused on separate domains (Sections 2.3, 2.5, 2.6, 2.12 and 2.14) or partially linked domains (Section 2.12.5). To the best of the author's knowledge, no research has focused on all five domains for the purpose of aiding the design decisions as well as capturing and representing the design knowledge as a map that links the requirements domain, design domain, manufacturing domain and cost domain.

In the introduction chapter of this thesis three research questions have been stated. The following paragraphs provide answers to these questions together with the concluding remarks.

Research question 1

Is it possible to create a knowledge map that captures the design process knowledge

through the dependency links between the requirements domain, design domain, manufacturing domain and cost domain?

Answer: the short answer to this question is *yes*. The concept of the proposed methodology framework has been based on the network interdependency model where links between the nodes serve as information carriers allowing the data to be fed in a forward or backward direction. The links between the requirements, design and cost domains capture the relational dependency information as well as the domain data which has been used for calculation and visualisation purposes. The framework allows to bring the cost data forward into the requirements domain by calculating the manufacturing cost of the features that satisfy a particular requirement. The overall impact of each requirement is also calculated and displayed in the requirements domain. Furthermore, knowledge that resides in the part as well as the design process knowledge, across the requirements, design and cost domains, has been captured as a result. The implementation of the methodology has been achieved using the BOXARR software which is based on the yFiles paradigm with enhanced computational functionality for system modelling, graph analysis and simulation. Furthermore, the embedded visualization and search algorithms have provided a powerful graph modelling capability.

Research question 2

Is it possible to automate the knowledge capture process? If yes, to what degree can the knowledge capture be automated?

Answer: the approach to answer this research question has been by investigating the types of knowledge and the knowledge capture approaches in the field of engineering. The literature survey revealed that the majority of design knowledge capture tools are generic prescriptive in nature. This means that the prescriptive approach is bound and restricted by the method used, therefore is too restrictive and inflexible. On the other hand, although the descriptive approach does not impose any restrictions on the design and decision, it might be too general. Both approaches, however, require human intervention. In the prescriptive approach, one has to follow the prescribed method which might become a burden. In the descriptive approach, one has to interpret and condense large amounts of data which, again, might become a burden. Therefore, the ideal approach would be to capture the design knowledge in an automated way without interrupting the engineers work flow. The proposed methodology framework captures the targeted design knowledge in an automated way. This has been achieved by embedding the code into each of the tools that are used in the main four product dev-

elopment domains (e.g. requirements, design, manufacturing and cost). The code captures targeted design data and saves it into the tabular data format. The full automation of the data capture has been achieved. However, due to time constraints this research focused only on the functional requirements. Further improvements, to expand the scope of the methodology that includes the non-functional requirements, have been suggested for future work in Section 7.2. The design knowledge has been captured from the following data sources (i.e. tools):

- DOORS - requirements data,
- NX CAD - design feature data,
- Vanguard Studio - manufacturing and cost data.

Research question 3

Will this knowledge map be useful for the design community and in what way?

Answer: the methodology has been applied to three case studies to test the usability and identify the benefits and shortcomings.

The first case study looked at the rear lock plate. Two scenarios have been investigated. First, the design knowledge has been captured using the proposed methodology. Second, the methodology was used to aid the decision making during the redesign activity of the rear lock plate. The first scenario has helped to understand the tools used in each of the domains and to identify the data that has been used to create the design knowledge map. The knowledge capture automation has been achieved for cost domain only. The case study revealed that some minor adjustments in some areas of the product development process have been necessary in order to achieve the knowledge capture automation. The identified areas were the design domain and requirements domain. The absence of the standard feature names and not being able to distinguish between the design features in the CAD tool made it difficult to automate the geometry data capture. The grouping of the modelling features that define the design feature in the CAD tool has been proposed by the author. Two immediate benefits have been observed. First, this type of approach allowed to define the geometry of the feature and standardise the naming of the feature. Second, having the modelling features, that represent a standard design feature, grouped in the CAD tool, de-cluttered the component modelling history tree and enabled the automated features data capture. In the first case study, the links between the requirements and features have been created manually. To automate this step the DFMEA tool has been used. However, an extra two columns to accommodate the requirements ID and requirements description were necessary in order to establish the link between the feature and the requirement and to achieve the automation.

The second scenario has explored the benefits of the methodology by applying it to the redesign activity of the rear lock plate. The design knowledge map of the rear lock plate proved to be very useful in identifying the potential cost reduction areas. The established links between the requirements, design, and cost domains provided the meaningful insight into the design of the rear lock plate and highlighted the high cost features. Following the relational links between the requirements and features the requirements interdependency has been established. This has allowed to identify the requirements that might be affected by the change of the feature design or by the change of any of the requirements. The fact that the most important design data (i.e. requirements, design features, and component unit cost data) has been previously captured in the relational dependency graph, the cost reduction scenarios have been simulated in a relatively short period of time. The manufacturing cost has been reduced by 16.5% and the overall component cost has been reduced by 6%. In summary, the methodology and BOXARR's graph modelling platform provide the means to run 'what if' scenarios and simulations more efficiently during the trade-off studies and cost optimisation activities. However, the feedback received from the engineers after the redesign activity of the rear lock plate suggested to test the calculated requirements impact values on more components.

The second case study tested the proposed methodology on a more complex component (e.g. IPT Stub Shaft). It has demonstrated that it is possible to capture the design knowledge without any significant interference in the product development work flow once the proposed improvements have been implemented. In addition, it proved that no extra work is required for more geometrically complex components. This case study has also verified that requirements impact can highlight the requirements that are linked to the high cost features as well as highlighting the number of interdependencies between the requirements.

The third case study has investigated how the requirements impact changes going from a single component system (i.e. stub shaft) to a system of components (triple seal flange joint sub-system). It has confirmed that the cost of the features, the number of features that the requirement is linked to and the interdependency between the requirements are reflected in the requirements impact value. In addition, the transition aspect from a single system to a sub-system is also captured and reflected in the requirements impact. However, it was observed that the requirements impact is highly dependent on the values of the *Weight on Features Cost (FCW)*, the *Weight on Features Complexity (FC)* (i.e. requirements interdependency) and the *Weight on Requirements Complexity (RC)* (i.e. number of links to features). Therefore, the full set of the design data (e.g. data from all the component requirements documents, all the components geometric (i.e. features) data as well as the cost data that is a part of the sub-system or system) has to be used in order to get the meaningful requirements impact value. In the third case study, however, only a mini sub-system (of three components) has been investigated, which is part of a larger IP Turbine sub-system. Additionally, only one

requirements document has been used for analysis. Thus, the value of the requirements impact of the triple seal flange joint mini sub-system does not represent the true requirements impact value of the IP Turbine sub-system. The feedback received from the engineers also expressed concern that the results might be distorted due to the fact that not a full data set has been used in the analysis. The evidence from the last case study suggests that the same effort is required applying the methodology on one component as well as on the mini sub-system of three components (once the framework is set up). However, the scalability aspect should be further investigated on larger sub-systems, ideally with components of varying complexity. Therefore, further research is necessary in this area with the suggestions from the author detailed in Section 7.2.

As a final conclusion, the proposed methodology together with the modelled graph provides a holistic view of the relational dependencies between the requirements and how they are linked to the features and manufacturing process costs. Better understanding of the relational complexities and interactions between the domains leads to better decision making. The fact that requirements, design features and cost information is in one place eliminates the need to go through many documents to search for the relevant information, saving time for the new projects as a result. Furthermore, the proposed methodology follows the work flow of the product development life cycle and does not require human intervention in order to capture the design knowledge. It is done automatically. The proposed methodology framework has achieved the aim of this research, which was to capture the design knowledge without any human intervention and to present it as a knowledge map with the relational links between the requirements, design and cost domains. However, there are limitations to all research. The following are the main limitations identified by the author:

- Although the design knowledge capture is an automated process, the upload of the captured knowledge into the BOXARR software, in order to create the knowledge map, has to be done manually.
- This research only looked at the functional requirements and had not investigated the effect of non-functional requirements.
- The study has only examined the individual components and a mini sub-system. The three case studies did not produce a sufficient data set that would confirm the scalability of the methodology.
- The research has been conducted in the aerospace industry. No evidence has been gathered to establish the validity of the methodology in other industries.

The following section provides the suggestions how to overcome the limitations and improve the methodology.

7.2 Future Work

This section discusses the areas that have not been implemented or explored due to the time limit of the EngD research program and provides few suggestions for future research in order to improve the current methodology.

As mentioned in the previous section, this research has only looked at the functional requirements since these often are the major design drivers. However, the non-functional requirements also shape or influence the design of the component. Therefore, further studies on how much non-functional requirements drive the product development process could be conducted. Also further work is required to understand what is the data overlap between the non-functional requirements and associated features and how (or if at all) it can be used to create links between them in an automated way.

Automating the upload function of the captured design knowledge data into BOXARR would further increase the efficiency of the methodology. Currently all the design knowledge data from the different sources is saved in the separate CSV files. Consolidating the data into one data file would further reduce the overall data upload and analysis time. The grouping of the modelling features, that define the geometry of a design feature, could be automated taking the burden from the designers.

More empirical studies are needed where the proposed methodology could be applied to the large sub-systems, such as IP Turbine, for example, or systems in order to understand the behaviour of the requirements impact and to establish its validity. BOXARR's capability to tackle complexity should provide real benefit when applying the methodology to the large sub-systems or systems. The methodology should also be applied in other industries that are involved in the product development (e.g. automotive) in order to understand the scope and applicability.

Appendices

Appendix A

The code (written in *Visual Basic (VB)* language) is embedded into the NX. Its objective is to generate the component's modelling features relational data and to output in the tabular data format. Firstly, the code identifies all the modelling features in the NX modelling history tree and saves them as objects with their names and unique IDs. Secondly, it identifies the relations between the modelling features and categorises them as parents and children. This data will be used to draw a relational graph in BOXARR. Thirdly, the data is saved into the CSV file format in the predefined location.

```
1 'File name - ListParentChildRelationshipV2.vb
2 'Generates CSV files with the objects and relational data together with
   the relevant metadata.
3
4 Option Strict Off
5 Imports System
6 Imports NXOpen
7 Imports NXOpen.UF
8 Imports NXOpen.Features
9 Imports System.IO
10
11 Module ListParentChildRelationship
12 Dim FilePath As String = "Z:\Rolls Royce\EngD Research\nxopen"
13 Dim FnName As String
14
15 Structure mFeatrureRelationship
16 Dim Fname As String
17 Dim Name As String
18 Dim isInt As Boolean
19 Dim fRelation() As mSecondLevelFeature
20 End Structure
21 Structure mSecondLevelFeature
22 Dim Fname As String
23 Dim Name As String
24 Dim IsRelationInternal As Boolean
25 End Structure
26
27 Private mTrackChild() As mFeatrureRelationship
28 Private mTrackParent() As mFeatrureRelationship
29
30
```

```
31 Sub Main()  
32 Try  
33 mGetChildren()  
34 mGetParents()  
35 SaveChildCSV()  
36  
37 SaveNameCSV()  
38 MsgBox("Data Saved!")  
39 Catch ex As Exception  
40 MsgBox(ex.Message)  
41  
42 End Try  
43 End Sub  
44  
45 Private Sub mGetChildren()  
46 ReDim mTrackChild(-1)  
47 Dim theSession As Session = Session.GetSession()  
48 Dim theUfSession As UfSession = UfSession.GetUfSession  
49 Dim workPart As Part = theSession.Parts.Work  
50 FnName = IO.Path.GetFileNameWithoutExtension(workPart.FullPath)  
51  
52 For Each mFeature As Feature In theSession.Parts.Work.Features  
53  
54 If mFeature.GetAllChildren().Length > 0 Then  
55  
56 ReDim Preserve mTrackChild(UBound(mTrackChild) + 1)  
57 Dim CF As Integer = UBound(mTrackChild)  
58 mTrackChild(CF).Fname = mFeature.GetFeatureName()  
59 mTrackChild(CF).Name = mFeature.Name  
60 mTrackChild(CF).isInt = mFeature.IsInternal  
61 ReDim mTrackChild(CF).fRelation(mFeature.GetAllChildren().Length - 1)  
62  
63 Dim i As Integer = 0  
64 For Each mChFeature As Feature In mFeature.GetAllChildren()  
65 Dim a1 As New mSecondLevelFeature  
66 a1.Fname = mChFeature.GetFeatureName()  
67 a1.IsRelationInternal = mChFeature.IsInternal  
68 mTrackChild(CF).fRelation(i).Fname = mChFeature.GetFeatureName()  
69 mTrackChild(CF).fRelation(i).name = mChFeature.Name  
70 mTrackChild(CF).fRelation(i).IsRelationInternal = mChFeature.IsInternal  
71 i = i + 1  
72 Next  
73  
74 End If  
75  
76 Next  
77 End Sub  
78  
79 Private Sub mGetParents()  
80 ReDim mTrackParent(-1)  
81 Dim theSession As Session = Session.GetSession()
```

```

82 Dim theUfSession As UfSession = UfSession.GetUfSession
83 Dim workPart As Part = theSession.Parts.Work
84 FnName = IO.Path.GetFileNameWithoutExtension(workPart.FullPath)
85
86 For Each mFeature As Feature In theSession.Parts.Work.Features
87
88 If mFeature.GetParents.Length > 0 Then
89
90 ReDim Preserve mTrackParent(UBound(mTrackParent) + 1)
91 Dim CF As Integer = UBound(mTrackParent)
92 mTrackParent(CF).Fname = mFeature.GetFeatureName()
93 mTrackParent(CF).Name = mFeature.Name
94 mTrackParent(CF).isInt = mFeature.IsInternal
95
96 ReDim mTrackParent(CF).fRelation(mFeature.GetParents().Length - 1)
97 Dim i As Integer = 0
98
99 For Each mChFeature As Feature In mFeature.GetParents()
100 Dim a1 As New mSecondLevelFeature
101 a1.Fname = mChFeature.GetFeatureName()
102 a1.Name = mChFeature.Name
103 a1.IsRelationInternal = mChFeature.IsInternal
104 mTrackParent(CF).fRelation(i) = a1
105 i = i + 1
106 Next
107
108 End If
109
110 Next
111 End Sub
112
113 Private Sub SaveChildCSV()
114 Dim FileName As String = FilePath & "\" & FnName & " - ParentChildren
    Arrows.csv"
115
116 Dim stream_writer As New IO.StreamWriter(FileName, False)
117
118 stream_writer.Write("Upstream Box (External Id), Downstream Box (External
    Id), Arrow Type" & vbCrLf)
119
120 Dim i, j As Integer
121 For i = 0 To UBound(mTrackChild)
122 For j = 0 To UBound(mTrackChild(i).fRelation)
123 If InStr(mTrackChild(i).Fname, "Feature Group") > 0 And InStr(mTrackChild
    (i).fRelation(j).Fname, "Feature Group") > 0 Then
124 stream_writer.Write(FnName & " - " & mTrackChild(i).Fname & " " &
    mTrackChild(i).Name & ", " & FnName & " - " & mTrackChild(i).fRelation
    (j).Fname & " " & mTrackChild(i).fRelation(j).Name & ", " & "Default"
    & vbCrLf)
125 ElseIf InStr(mTrackChild(i).Fname, "Feature Group") > 0 Then

```



```

126 stream_writer.Write(FnName & " - " & mTrackChild(i).Fname & " " &
    mTrackChild(i).Name & ", " & FnName & " - " & mTrackChild(i).fRelation
    (j).Fname & ", " & "Default" & vbCrLf)
127 ElseIf InStr(mTrackChild(i).fRelation(j).Fname, "Feature Group") > 0 Then
128 stream_writer.Write(FnName & " - " & mTrackChild(i).Fname & ", " & FnName
    & " - " & mTrackChild(i).fRelation(j).Fname & " " & mTrackChild(i).
    fRelation(j).Name & ", " & "Default" & vbCrLf)
129 Else
130 stream_writer.Write(FnName & " - " & mTrackChild(i).Fname & ", " & FnName
    & " - " & mTrackChild(i).fRelation(j).Fname & ", " & "Default" &
    vbCrLf)
131 End If
132
133 Next
134 Next
135 stream_writer.Close()
136 End Sub
137
138 Private Sub SaveNameCSV()
139 Dim FileName As String = FilePath & "\" & FnName & " - ParentChildren
    Boxes.csv"
140
141 Dim stream_writer As New IO.StreamWriter(FileName, False)
142
143 stream_writer.Write("External Id, Name" & vbCrLf)
144
145 Dim NList(0) As String
146 Dim NListF(0) As String
147
148 Dim i, j, k As Integer
149
150 NList(0) = mTrackChild(0).Fname
151 NListF(0) = mTrackChild(0).Name
152 Dim MatchFound As Boolean
153
154 For i = 0 To UBound(mTrackChild)
155 MatchFound = False
156 For k = 0 To UBound(NList)
157 If NList(k) = mTrackChild(i).Fname Then
158 MatchFound = True
159 Exit For
160 End If
161 Next
162
163 If MatchFound = False Then
164 ReDim Preserve NList(UBound(NList) + 1)
165 NList(UBound(NList)) = mTrackChild(i).Fname
166 ReDim Preserve NListF(UBound(NListF) + 1)
167 NListF(UBound(NListF)) = mTrackChild(i).Name
168 End If
169

```

```
170 For j = 0 To UBound(mTrackChild(i).fRelation)
171
172 MatchFound = False
173 For k = 0 To UBound(NList)
174 If NList(k) = mTrackChild(i).fRelation(j).Fname Then
175 MatchFound = True
176 Exit For
177 End If
178 Next
179
180 If MatchFound = False Then
181 ReDim Preserve NList(UBound(NList) + 1)
182 NList(UBound(NList)) = mTrackChild(i).fRelation(j).Fname
183 ReDim Preserve NListF(UBound(NListF) + 1)
184 NListF(UBound(NListF)) = mTrackChild(i).fRelation(j).Name
185 End If
186
187 Next
188 Next
189
190 For i = 0 To UBound(NList)
191 If InStr(NList(i), "Feature Group") > 0 Then
192 stream_writer.Write(FnName & " - " & NList(i) & " " & NListF(i) & ", " &
    NListF(i) & vbCrLf)
193 Else
194 stream_writer.Write(FnName & " - " & NList(i) & ", " & NList(i) & vbCrLf)
195 End If
196
197 Next
198 stream_writer.Close()
199
200 End Sub
201 End Module
```

Appendix B

The code (written in *VB* language) is embedded into the NX. It runs through the modelling tree in the NX and identifies the existing feature groups. The code looks for the ‘*Design Features*’ group. The design feature names are stored in the ‘*Design Features*’ group as *Extracted Body* objects that represent the feature. All the design feature names present in the ‘*Design Features*’ group are saved into the CSV file, in a specified location, together with its ‘External Id’, ‘Box Type’ and ‘WBS Element (External Id)’ as metadata.

```

1 'File name - features information V1.vb
2 'Generates CSV file with the design feature names and the relevant
   metadata.
3
4 Option Strict Off
5 Imports System
6 Imports System.Collections.Generic
7 Imports NXOpen
8 Imports NXOpen.UF
9
10
11 Module Module2
12
13 Dim FilePath As String = "Z:\Rolls Royce\EngD Research\Journal Paper\
14 Design Features and Material Mass"
15 Dim FnName As String
16
17 Structure FeatrureAttributes
18 Dim Fname As String
19 End Structure
20
21 Private CalcAttr() As FeatrureAttributes
22
23 Sub Main()
24
25 Dim iBodies(0) As IBody
26 Dim ufs As NXOpen.UF.UFSession = NXOpen.UF.UFSession.GetUFSession()
27 Dim theSession As Session = Session.GetSession()
28 Dim theUfSession As UFSession = UFSession.GetUFSession
29
30 If IsNothing(theSession.Parts.Work) Then
31 Return
32 End If
33
34 Dim workPart As Part = theSession.Parts.Work
35 Dim lw As ListingWindow = theSession.ListingWindow
36 lw.Open()
37
38 Const undoMarkName As String = "NXJ feature group"

```

```

39 Dim markId1 As Session.UndoMarkId
40 markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible,
    undoMarkName)
41
42 Dim materialsInPart As New List(Of PhysicalMaterial)
43 Dim unusedMaterialsInPart As New List(Of String)
44
45 For Each tempMaterial As PhysicalMaterial In
46 workPart.MaterialManager.PhysicalMaterials.GetUsedMaterials
47 materialsInPart.Add(tempMaterial)
48 unusedMaterialsInPart.Add(tempMaterial.Name)
49 Next
50
51 FnName = IO.Path.GetFileNameWithoutExtension(workPart.FullPath)
52
53 dim cr as integer
54 cr =0
55
56 For Each tempFeat As Features.Feature In workPart.Features
57
58 If TypeOf (tempFeat) Is Features.FeatureGroup Then
59 Dim numFeatures As Integer
60 Dim setFeatureTags() As Tag
61 theUfSession.Modl.AskAllMembersOfSet(tempFeat.Tag, setFeatureTags,
    numFeatures)
62 Dim setFeatures As New List(Of Features.Feature)
63
64 'get features from tags
65 For Each tempTag As Tag In setFeatureTags
66 setFeatures.Add(Utilities.NXObjectManager.Get(tempTag))
67 Next
68
69 if tempFeat.Name = "Design Features" Then
70
71 For Each temF As Features.Feature In tempFeat.GetParents
72 Redim Preserve CalcAttr(cr)
73
74 try
75 CalcAttr(cr).Fname = temF.Name
76
77 catch ex as exception
78 CalcAttr(cr).Fname = temF.Name
79
80 end try
81
82 cr=cr+1
83 Next
84 end if
85 End If
86
87 Next

```

```
88
89 lw.WriteLine("Design Features")
90 lw.WriteLine(" ")
91
92 Dim i As Integer
93
94 For i = 0 To UBound(CalcAttr)
95 lw.WriteLine(CalcAttr(i).Fname)
96 Next
97
98 lw.Close()
99 SavetoCSV()
100
101 End Sub
102
103
104 Private Sub SavetoCSV()
105
106 Dim FileName As String = FilePath & "\" & FnName & " - Design Features.
    csv"
107
108 Dim stream_writer As New IO.StreamWriter(FileName, False)
109
110 Dim NameCut as string
111 if instr(FnName, "-") > 1 then
112 NameCut = FnName.Substring(0, FnName.IndexOf("-"))
113 else
114 NameCut = FnName
115 end if
116
117 stream_writer.Write("External Id,Name,Box Type,WBS Element (External Id)"
    & vbCrLf)
118
119 Dim i As Integer
120 For i = 0 To UBound(CalcAttr)
121
122 stream_writer.Write(NameCut & "-" & CalcAttr(i).Fname & "," & CalcAttr(i)
    .Fname & ","
123 & "Feature" & "," & NameCut & "-Design Features" & vbCrLf)
124 Next
125
126 stream_writer.Close()
127 End Sub
128
129 Public Function GetUnloadOption(ByVal dummy As String) As Integer
130
131 'Unloads the image immediately after execution within NX
132 GetUnloadOption = NXOpen.Session.LibraryUnloadOption.Immediately
133
134 End Function
135 End Module
```

Appendix C

The code (written in *DScript* language) is embedded into the Vanguard. There are three functions separated by stars in the code below. The first function '*Value Streams*' loops through the nodes and saves the following data into a structured tabular format: value stream name (i.e. '*Value Streams*'), cost of the value stream (i.e. '*Value*'), unique object ID (i.e. '*External Id*'), object data type (i.e. '*Box Type*'), sub group unique ID (i.e. '*Parent External Id*'), group unique ID (i.e. '*Parent WBS External Id*') and the relationship type (i.e. '*Arrow Type*'). The second function '*WBS Elements*', loops through the Process cost node and saves the group OP Name, the group OP Number, the unique group OP ID and the unique sub group OP ID into the tabular data format. The third function '*Return relational Data*', loops through the nodes and saves the relevant cost data and the data needed to create the relational links between the features and manufacturing OP sequences into the structured tabular data format.

```

1  "### Vanguard code that generates the Value Stream data ###";
2
3  Value Streams:={
4  "### Initiallise Data Table ###";
5  Data4_hlegend=[
6  "Value Streams",
7  "Value",
8  "External Id",
9  "Box Type",
10 "Parent External Id",
11 "Parent WBS External Id",
12 "Part Box External Id",
13 "Arrow Type"];
14
15 nodenames:=[
16 "Raw material cost",
17 "Procured parts cost",
18 "Procured service cost",
19 "Consumables cost",
20 "Cost of non quality",
21 "Overheads and other cost",
22 "Profit"];
23
24 Data4[0]=each(x,x,nodenames);
25 Data4[1]=each(eval(x)/£,x,nodenames);
26 Data4[2]=each(Part number+"-"+x,x,nodenames);
27 Data4[3]=each("Value Stream",x,nodenames);
28 Data4[4]=each(Part number+"-"+x+"-"+Parent,x,nodenames);
29 Data4[5]=each("Value Streams"+"-"+Part number,x,nodenames);
30 Data4[6]=each(Part number+"-COS",x,nodenames);
31 Data4[7]=each("Value Stream to Final Cost",x,nodenames);}
32 "###*****###";

```

```

33
34 "### Vanguard code that generates the OPs WBS Elements ###";
35
36 WBS Elements:={
37 "### Initiallise Data Table ###";
38 Data2_hlegend=[
39 "OP Number",
40 "Name",
41 "External Id",
42 "Parent (External Id)"];
43
44 var ProcessCat;
45 ProcessCat=[
46 "Other",
47 "Chemical",
48 "Inspection Other",
49 "Assembly",
50 "Bench",
51 "Coating",
52 "Inspection Measurement",
53 "Inspection NDE",
54 "Thermal",
55 "Surface Modification"];
56
57 var RunTimeUnscaledNodes=user(all("Op * total cost"));
58 var FeatureOpNodes=[];
59 var WBSElement=[];
60 var WBSElementID=[];
61 var ParentWBSElementID=[];
62 var TempString,NodePrefix,TempNodeName,Process;
63
64 each({
65 "### Get node prefix = Op number ###";
66 NodePrefix=GetFirstChars(x,7);
67
68 "### Add Op number to list ###";
69 FeatureOpNodes!=NodePrefix;
70
71 "### Get node prefix = Op number + String ###";
72 NodePrefix=GetFirstChars(x,7);
73 WBSElement!=NodePrefix+" Sub Ops";
74 WBSElementID!=eval("Part number")+"-"+NodePrefix;
75 TempString=GetCommentLine(x,1);
76 var indx=index(ProcessCat,TempString);
77 if(indx!=-1)
78 Process=eval("Part number")+"-Additional costs";
79 else
80 Process=eval("Part number")+"-Machining costs";
81 ParentWBSElementID!=Process;},x,RunTimeUnscaledNodes);
82
83 Data2[0]=FeatureOpNodes;

```

```

84 Data2[1]=WBSElement;
85 Data2[2]=WBSElementID;
86 Data2[3]=ParentWBSElementID;}
87 "###*****";
88
89 "### Vanguard code that generates the cost data and the data needed to
90 create relational links between features and manufacturing OP
91 sequence###";
92
93 Return relational Data:={
94 "### Initiallise Data Table ###";
95 Data3_hlegend=[
96 "OP Number",
97 "Op Description",
98 "Process",
99 "Name",
100 "External Id",
101 "Upstream Box",
102 "Run Time",
103 "Cost Rate",
104 "OP Cost",
105 "WBS Element (External Id)",
106 "Box Type",
107 "Arrow Type",
108 "Upstream Box (External Id)",
109 "Part Box Arrow Type"];
110
111 var RunTimeUnscaledNodes=user(all("Op *_* run time"));
112 var FeatureOpNodes=[];
113 var FeatureList=[];
114 var FeatureSubOpNodes=[];
115 var FeatureSubOpNodesExId=[];
116 var FeatureOpTimes=[];
117 var FeatureOpRates=[];
118 var OpCost=[];
119 var OpDescription=[];
120 var ProcessCategory=[];
121 var WBSElement=[];
122 var BoxType=[];
123 var ArrowType=[];
124 var PartBoxExtId=[];
125 var PrtBArrowType=[];
126 var TempString,NodePrefix,TempNodeName,ExtId;
127 each({
128 "### Get node prefix = Op number ###";
129 NodePrefix=GetFirstChars(x,7);
130
131 "### Add Op number to list ###";
132 FeatureOpNodes!=NodePrefix;
133
134 "### Get node comment ###";

```



```
135 TempString=_USEREVAL("document."+x+".note");
136
137 "### If no comment, set to n/a so that list gets resized properly ###";
138 if(TempString==null)
139 TempString=eval("Part number")+ "-" + "NVA Costs";
140
141 "### Add comment to list ###";
142 FeatureList!=TempString;
143
144 "### Add node name to list ###";
145 FeatureSubOpNodes!=GetFirstChars(x,10);
146
147 "### Add node name external Id to list ###";
148 ExtId=GetFirstChars(x,10);
149 FeatureSubOpNodesExId!=eval("Part number")+ "-" + ExtId;
150
151 "### Add node value to list ###";
152 FeatureOpTimes!=_USEREVAL(x);
153
154 "### Get cost rate node name ###";
155 TempNodeName=NodePrefix+" run equipment cost rate";
156
157 "### Add cost rate to list ###";
158 FeatureOpRates!=_USEREVAL(TempNodeName);
159
160 "### Get main op node name ###";
161 TempNodeName=NodePrefix+" run time";
162
163 "### Get comment 1st line = category ###";
164 TempString=GetCommentLine(TempNodeName,1);
165
166 "### If no category, set to n/a so that list gets resized properly ###";
167 if(TempString==null)
168 TempString=eval("Part number")+ "-" + "NVA Costs";
169
170 "### Add category to list ###";
171 ProcessCategory!=TempString;
172
173 "### Get comment 2nd line = descriptioncategory ###";
174 TempString=GetCommentLine(TempNodeName,2);
175
176 "### If no category, set to n/a so that list gets resized properly ###";
177 if(TempString==null)
178 TempString=eval("Part number")+ "-" + "NVA Costs";
179
180 "### Add description to list ###";
181 OpDescription!=TempString;
182 NodePrefix=GetFirstChars(x,7);
183 WBSElement!=eval("Part number")+ "-" + NodePrefix;
184 BoxType!="OP";
185 ArrowType!="Features to Op Sequence";
```

```
186 PartBoxExtId!=eval("Part number")+"-COS";
187 PrtBArrowType!="Value Stream to Final Cost";},x,RunTimeUnscaledNodes);
188
189 Data3[0]=FeatureOpNodes;
190 Data3[1]=OpDescription;
191 Data3[2]=ProcessCategory;
192 Data3[3]=FeatureSubOpNodes;
193 Data3[4]=FeatureSubOpNodesExId;
194 Data3[5]=FeatureList;
195 Data3[6]=FeatureOpTimes;
196 Data3[7]=FeatureOpRates;
197 Data3[8]=FeatureOpRates*FeatureOpTimes/£;
198 Data3[9]=WBSElement;
199 Data3[10]=BoxType;
200 Data3[11]=ArrowType;
201 Data3[12]=PartBoxExtId;
202 Data3[13]=PrtBArrowType;}
```

Appendix D

The code (written in VB language) is embedded into the DFMEA excel based tool. When executed it prompts the user to select the component features and requirements CSV files (that have been created earlier in the process) and uploads the features and requirements data into the DFMEA tool.

```

1 Option Explicit
2
3 Sub ImportDesignFeatures()
4
5 Dim msgRetVal As Integer
6 msgRetVal = MsgBox("The Data in Sheet " & "Design Features List" & " will be
    deleted!
7 OK to proceed?", vbOKCancel)
8 If msgRetVal = 1 Then
9 ' Get customer workbook...
10 Dim customerBook As Workbook
11 Dim filter As String
12 Dim caption As String
13 Dim customerFilename As String
14 Dim customerWorkbook As Workbook
15 Dim targetWorkbook As Workbook
16
17 Sheets("Design Features List").Cells.Clear
18
19 ' make weak assumption that active workbook is the target
20 Set targetWorkbook = Application.ActiveWorkbook
21
22 ' get the customer workbook
23 filter = "Text files (*.csv),*.csv"
24 caption = "Please Select an input file "
25 customerFilename = Application.GetOpenFilename(filter, , caption)
26
27 Set customerWorkbook = Application.Workbooks.Open(customerFilename)
28
29 'Copy "Design Features List" from the source file
30 Dim targetSheet As Worksheet
31 Set targetSheet = targetWorkbook.Worksheets("Design Features List")
32 Dim sourceSheet As Worksheet
33 Set sourceSheet = customerWorkbook.Worksheets(1)
34
35 Dim StR, EnR, StC, EnC, Sh2R, i As Integer
36 Dim rn As Range
37
38 Set rn = sourceSheet.UsedRange
39
40 StR = 1
41 EnR = rn.Rows.Count + rn.Row - 1

```

```

42 StC = 1
43 EnC = 2
44 Sh2R = 1
45
46 For i = StR To EnR
47     targetSheet.Cells(Sh2R, 1) = sourceSheet.Cells(i, 1)
48     targetSheet.Cells(Sh2R, 2) = sourceSheet.Cells(i, 2)
49     Sh2R = Sh2R + 1
50 Next i
51 Sh2R = Sh2R + 1
52
53 ' Close customer workbook
54 customerWorkbook.Close
55 End If
56 End Sub
57
58 Option Explicit
59 Sub ImportRequirements()
60
61     Dim msgRetVal As Integer
62     msgRetVal = MsgBox("The Data in Sheet " & "Requirements" & " will be deleted!"
63         & "OK to proceed?", vbOKCancel)
64     If msgRetVal = 1 Then
65
66         ' Get customer workbook...
67         Dim customerBook As Workbook
68         Dim filter As String
69         Dim caption As String
70         Dim customerFilename As String
71         Dim customerWorkbook As Workbook
72         Dim targetWorkbook As Workbook
73
74         Sheets("Requirements").Cells.Clear
75
76         ' make weak assumption that active workbook is the target
77         Set targetWorkbook = Application.ActiveWorkbook
78         ' get the customer workbook
79         filter = "Text files (*.csv),*.csv"
80         caption = "Please Select an input file "
81         customerFilename = Application.GetOpenFilename(filter, , caption)
82
83         Set customerWorkbook = Application.Workbooks.Open(customerFilename)
84
85         'Copy "Design Features List" from the source file
86         Dim targetSheet As Worksheet
87         Set targetSheet = targetWorkbook.Worksheets("Requirements")
88         Dim sourceSheet As Worksheet
89         Set sourceSheet = customerWorkbook.Worksheets(1)
90
91         Dim StR, EnR, StC, EnC, Sh2R, i As Integer
92         Dim rn As Range

```

```
93 targetSheet.Cells(1, 1).Value = "Requirement Id"
94 targetSheet.Cells(1, 2).Value = "Description"
95
96 Set rn = sourceSheet.UsedRange
97
98 StR = 2
99 EnR = rn.Rows.Count + rn.Row - 1
100 StC = 1
101 EnC = 2
102 Sh2R = 2
103
104 For i = StR To EnR
105 targetSheet.Cells(Sh2R, 1) = sourceSheet.Cells(i, 1)
106 targetSheet.Cells(Sh2R, 2) = sourceSheet.Cells(i, 2)
107 Sh2R = Sh2R + 1
108 Next i
109 Sh2R = Sh2R + 1
110
111 ' Close customer workbook
112 customerWorkbook.Close
113 End If
114 End Sub
```

Appendix E

The code (written in *VB* language) is embedded into the DFMEA excel based tool. This code is executed after the DFMEA analysis is completed. It prompts the user to select the file location and saves the requirements and features relationship data into CSV file format. This data will be used in BOXARR to create links between the requirements and design features.

```

1 Option Explicit
2 Dim FeatList() As String
3 Dim newFeatList As Boolean
4
5 Sub ExportLinks()
6
7 Dim FilePath As String
8 FilePath = "Z:\Rolls Royce\EngD Research\Rotatives\Shafts\IPT Shaft\
9 Boxarr Mapping\DFMEA Examples\Demo"
10 Dim FnName As String
11 Dim StR, EnR, StC, EnC, Sh2R, Sh2R2, i, J, K, L, C As Integer
12 Dim StCB, EnCB, ChEnR As Integer
13 Dim rn As Range
14 Dim ur As Range
15 Dim rCount As Integer
16
17 Sheets("Export Links").Cells.Clear
18
19 Set rn = Sheet2.UsedRange
20
21 'Copy data from "Sheet2"
22 StR = 25
23 EnR = rn.Rows.Count + rn.Row - 1
24 Sh2R = 2
25
26 Worksheets("Export Links").Range("A1").Value = "Upstream Box (External Id
27 )"
28 Worksheets("Export Links").Range("B1").Value = "Requirement"
29 Worksheets("Export Links").Range("C1").Value = "Design Feature"
30 Worksheets("Export Links").Range("D1").Value = "Downstream Box (External
31 Id)"
32 Worksheets("Export Links").Range("E1").Value = "Arrow Type"
33
34 rCount = CountingRows("Design Features List")
35
36 newFeatList = True
37
38 Dim FileName As String
39 FileName = FilePath & "\" & Sheet2.Cells(15, 1).Text & " - Req to Feat
40 Arrows.csv"
41 Open FileName For Output As #1

```

```

39
40 Print #1, "" & "Upstream Box (External Id)" & "";
41 Print #1, ",";
42 Print #1, "" & "Requirement" & "";
43 Print #1, ",";
44 Print #1, "" & "Design Feature" & "";
45 Print #1, ",";
46 Print #1, "" & "Downstream Box (External Id)" & "";
47 Print #1, ",";
48 Print #1, "" & "Arrow Type" & "";
49 Print #1,
50
51 For i = StR To EnR
52 If chkFeatList(Sheet2.Cells(i, 4) & ", " & Sheet2.Cells(i, 2)) = False
53     Then
54 Sheet10.Cells(Sh2R, 1) = Sheet2.Cells(i, 2)
55 Sheet10.Cells(Sh2R, 2) = Sheet2.Cells(i, 3)
56 Sheet10.Cells(Sh2R, 3) = Sheet2.Cells(i, 4)
57 Sheet10.Cells(Sh2R, 4) = lookUpDwg(Sheet2.Cells(i, 4), rCount)
58 Sheet10.Cells(Sh2R, 5) = "Requirements to Features"
59
60 Print #1, "" & Sheet10.Cells(Sh2R, 1).Text & "";
61 Print #1, ",";
62 Print #1, "" & Sheet10.Cells(Sh2R, 2).Text & "";
63 Print #1, ",";
64 Print #1, "" & Sheet10.Cells(Sh2R, 3).Text & "";
65 Print #1, ",";
66 Print #1, "" & Sheet10.Cells(Sh2R, 4).Text & "";
67 Print #1, ",";
68 Print #1, "" & Sheet10.Cells(Sh2R, 5).Text & "";
69 Print #1,
70 Sh2R = Sh2R + 1
71
72 End If
73 Next i
74
75 Close #1
76
77 MsgBox ("Data Exported Successfully")
78 End Sub
79
80 Function chkFeatList(ByVal lName As String) As Boolean
81     chkFeatList = False
82     Dim i As Integer
83
84     If newFeatList = True Then
85         ReDim FeatList(0)
86         FeatList(0) = lName
87         newFeatList = False
88     Else
89         For i = 0 To UBound(FeatList)
90             If lName = FeatList(i) Then

```

```

89 chkFeatList = True
90 Exit For
91 End If
92 Next i
93 If chkFeatList = False Then
94 ReDim Preserve FeatList(UBound(FeatList) + 1)
95 FeatList(UBound(FeatList)) = lName
96 End If
97 End If
98
99 End Function
100
101 Function lookUpDwg(ByVal lName As String, ByVal rCount As Integer) As
    String
102 Dim i As Integer
103 lookUpDwg = ""
104
105 For i = 1 To rCount
106 If DesignFeatList.Cells(i, 2) = lName Then
107
108 lookUpDwg = DesignFeatList.Cells(i, 1)
109 Exit For
110 End If
111 Next i
112
113 End Function
114
115 Sub QuoteCommaExport()
116 ' Dimension all variables.
117 Dim DestFile As String
118 Dim FileNum As Integer
119 Dim ColumnCount As Integer
120 Dim RowCount As Integer
121
122 ' Prompt user for destination file name.
123 DestFile = InputBox("Enter the destination filename" _
124 & Chr(10) & "(with complete path):", "Quote-Comma Exporter")
125
126 ' Obtain next free file handle number.
127 FileNum = FreeFile()
128
129 ' Turn error checking off.
130 On Error Resume Next
131
132 ' Attempt to open destination file for output.
133 Open DestFile For Output As #FileNum
134
135 ' If an error occurs report it and end.
136 If Err <> 0 Then
137 MsgBox "Cannot open filename " & DestFile
138 End

```



```

139 End If
140
141 ' Turn error checking on.
142 On Error GoTo 0
143
144 ' Loop for each row in selection.
145 For RowCount = 1 To Selection.Rows.Count
146
147 ' Loop for each column in selection.
148 For ColumnCount = 1 To Selection.Columns.Count
149
150 ' Write current cell's text to file with quotation marks.
151 Print #FileNum, """" & Selection.Cells(RowCount, _
152 ColumnCount).Text & """";
153
154 ' Check if cell is in last column.
155 If ColumnCount = Selection.Columns.Count Then
156 ' If so, then write a blank line.
157 Print #FileNum,
158 Else
159 ' Otherwise, write a comma.
160 Print #FileNum, ",";
161 End If
162 ' Start next iteration of ColumnCount loop.
163 Next ColumnCount
164 ' Start next iteration of RowCount loop.
165 Next RowCount
166
167 ' Close destination file.
168 Close #FileNum
169 End Sub
170
171 Public Function CountingRows(ByVal sheetName) As Integer
172
173 Dim sh As Worksheet
174 Dim rn As Range
175 Set sh = ThisWorkbook.Sheets(sheetName)
176
177 Dim K As Long
178
179 Set rn = sh.UsedRange
180 K = rn.Rows.Count + rn.Row - 1
181
182 CountingRows = K
183
184 End Function
185
186 Public Function CountingCols(ByVal sheetName) As Integer
187
188 Dim sh As Worksheet
189 Dim rn As Range

```

```
190 Set sh = ThisWorkbook.Sheets(sheetName)
191
192 Dim K As Long
193
194 Set rn = sh.UsedRange
195 K = rn.Columns.Count + rn.Column - 1
196
197 CountingCols = K
198
199 End Function
```

Appendix F

The code (written in *VB* language) is embedded into the NX. The primary objective of this code is to extract the feature volume, mass and material mass ratio from the Condition of Supply (e.g. forging, casting) part geometry and save it into the CSV file format. To achieve this the code loops through the design history tree in the NX and identifies that '*Design Features*' group exists. It then loops through all the design features and extracts the volume, mass and material mass ratio of the feature. If the feature is switched off, the message prompts the user to switch a particular feature on in order to extract the mass ratio. In addition, the code also displays the feature name, the material name, the volume, and the mass of each feature as well as the total mass of the component and the material mass ratio of the feature on the information window in the live NX session.

```

1  'File name - List of Design Features as Extracted Bodies_combined V1.vb
2
3  Option Strict Off
4  Imports System
5  Imports System.Collections.Generic
6  Imports NXOpen
7  Imports NXOpen.UF
8
9
10 Module Module2
11
12 Dim FilePath As String = "Z:\Rolls Royce\EngD Research\Journal Paper\
13 Design Features and Material Mass"
14 Dim FnName As String
15
16 Structure FeatrureAttributes
17 Dim Fname As String
18 Dim PMaterial As String
19 Dim FVolume As Single
20 Dim FMass As Single
21 Dim PartMass As Single
22 Dim AxialL As Single
23 Dim RadialL As Single
24 Dim ZAxisL As Single
25 End Structure
26
27 Private CalcAttr() As FeatrureAttributes
28 Dim mMass as Single
29 Sub Main()
30
31 Dim iBodies(0) As IBody
32 Dim ufs As NXOpen.UF.UFSession = NXOpen.UF.UFSession.GetUFSession()
33 Dim theSession As Session = Session.GetSession()
34 Dim theUfSession As UFSession = UFSession.GetUFSession

```

```

35
36 If IsNothing(theSession.Parts.Work) Then
37 'active part required
38 Return
39 End If
40
41 Dim workPart As Part = theSession.Parts.Work
42 Dim lw As ListingWindow = theSession.ListingWindow
43 lw.Open()
44
45 Const undoMarkName As String = "NXJ feature group"
46 Dim markId1 As Session.UndoMarkId
47 markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible,
    undoMarkName)
48
49 Dim materialsInPart As New List(Of PhysicalMaterial)
50 Dim unusedMaterialsInPart As New List(Of String)
51
52 For Each tempMaterial As PhysicalMaterial In
53 workPart.MaterialManager.PhysicalMaterials.GetUsedMaterials
54 materialsInPart.Add(tempMaterial)
55 unusedMaterialsInPart.Add(tempMaterial.Name)
56 Next
57
58 FnName = IO.Path.GetFileNameWithoutExtension(workPart.FullPath)
59
60 mMass = 0
61 dim cr as integer
62 cr =0
63
64 For Each tempFeat As Features.Feature In workPart.Features
65
66 If TypeOf (tempFeat) Is Features.FeatureGroup Then
67 Dim numFeatures As Integer
68 Dim setFeatureTags() As Tag
69 theUfSession.Modl.AskAllMembersOfSet(tempFeat.Tag, setFeatureTags,
    numFeatures)
70 Dim setFeatures As New List(Of Features.Feature)
71
72 'get features from tags
73 For Each tempTag As Tag In setFeatureTags
74 setFeatures.Add(Utilities.NXObjectManager.Get(tempTag))
75 Next
76
77 if tempFeat.Name = "Design Features" Then
78
79 For Each temF As Features.Feature In tempFeat.GetParents
80 Redim Preserve CalcAttr(cr)
81 Dim Supportbody As Body = CType(workPart.Bodies.FindObject(temF.
    JournalIdentifier),
82 Body)

```

```

83 Dim materialAttribute As NXObject.AttributeInformation
84 materialAttribute = Supportbody.GetUserAttribute("Material",
85 NXObject.AttributeType.String, -1)
86 unusedMaterialsInPart.Remove(materialAttribute.StringValue)
87
88 Dim massUnits1(4) As Unit
89 massUnits1(0) = CType(workPart.UnitCollection.FindObject("
    SquareMilliMeter"), Unit)
90 massUnits1(1) = CType(workPart.UnitCollection.FindObject("CubicMilliMeter
    "), Unit)
91 massUnits1(2) = CType(workPart.UnitCollection.FindObject("Kilogram"),
    Unit)
92 massUnits1(3) = CType(workPart.UnitCollection.FindObject("MilliMeter"),
    Unit)
93 massUnits1(4) = CType(workPart.UnitCollection.FindObject("Newton"), Unit)
94
95 Dim mBodies As New List(Of Body)
96 mBodies.Add(Supportbody)
97 dim bbox(5) as double
98 Dim measureBodies1 As MeasureBodies
99 measureBodies1 = workPart.MeasureManager.NewMassProperties(massUnits1,
    0.99,
100 mBodies.ToArray)
101
102 try
103 ufs.Modl.AskBoundingBox(Supportbody.tag, bbox)
104
105 CalcAttr(cr).Fname = temF.Name
106 CalcAttr(cr).PMaterial = materialAttribute.StringValue
107 CalcAttr(cr).FVolume = measureBodies1.Volume
108 CalcAttr(cr).FMass = measureBodies1.Mass
109 CalcAttr(cr).AxialL = math.abs(bbox(0)-bbox(3))
110 CalcAttr(cr).RadialL = math.abs(bbox(1)-bbox(4))
111 CalcAttr(cr).ZAxisL = math.abs(bbox(2)-bbox(5))
112 CalcAttr(cr).PartMass = mMass + measureBodies1.Mass
113 catch ex as exception
114 CalcAttr(cr).Fname = temF.Name
115 CalcAttr(cr).PMaterial = materialAttribute.StringValue
116 CalcAttr(cr).FVolume = measureBodies1.Volume
117 CalcAttr(cr).FMass = measureBodies1.Mass
118 CalcAttr(cr).PartMass = mMass + measureBodies1.Mass
119 end try
120
121 cr=cr+1
122 Next
123 end if
124 End If
125
126 Next
127
128 Dim i As Integer

```

```

129 Dim PMass as Integer
130 mMass = 0
131 dim isValid as boolean = true
132 For i = 0 To UBound(CalcAttr)
133 mMass = mMass +CalcAttr(i).FMass
134 if CalcAttr(i).ZAxisL =0 then isValid=false
135 next
136 For i = 0 To UBound(CalcAttr)
137
138 if CalcAttr(i).ZAxisL = 0 Then
139 Try
140 lw.WriteLine("
141      *****")
142 lw.WriteLine(" ")
143 lw.WriteLine(CalcAttr(i).Fname & " Feature suppressed!!! Total mass
144      cannot be calculated." )
145 lw.WriteLine(" ")
146 MsgBox(CalcAttr(i).Fname & " Feature suppressed!!! Total mass cannot be
147      calculated.")
148 Catch ex As Exception
149 MsgBox(ex.Message)
150 End Try
151 else
152
153 lw.WriteLine("
154      *****")
155 lw.WriteLine(" ")
156 lw.WriteLine(CalcAttr(i).Fname)
157
158 if len(CalcAttr(i).PMaterial) >0 then
159 lw.WriteLine("Material Assigned: " & CalcAttr(i).PMaterial)
160 lw.WriteLine(" ")
161 lw.WriteLine("Volume: " & Format(CalcAttr(i).FVolume, "0.000#") & " mm^3"
162      )
163 lw.WriteLine("Mass: " & Format(CalcAttr(i).FMass, "0.000#") & " kg")
164 if isValid then
165 lw.WriteLine("(Mass/Total Mass) Ratio: " & Format((CalcAttr(i).FMass/
166      mMass)*100, "0.0#")
167 & " %")
168 else
169 lw.WriteLine("Relative Mass is not Valid " )
170 end if
171 lw.WriteLine(" ")
172 lw.WriteLine("Axial Length: " & Format(CalcAttr(i).AxialL, "0.0#") & " mm
173      ")
174 lw.WriteLine("Radial Length: " & Format(CalcAttr(i).RadialL, "0.0#") & "
175      mm")
176 lw.WriteLine("Z Axis Length: " & Format(CalcAttr(i).ZAxisL, "0.0#") & "
177      mm")
178 lw.WriteLine(" ")

```

```

170 lw.WriteLine("
    *****")
171
172 else
173 try
174 lw.WriteLine("Material not Assigned! ")
175 lw.WriteLine(" ")
176 MsgBox("Material for " & CalcAttr(i).Fname & " is not Assigned!!!
177 Total mass cannot be calculated.")
178 Catch ex As Exception
179 MsgBox(ex.Message)
180 End Try
181 end if
182
183 length: " & math.abs(bbox(0)-bbox(3)) & " radial length: " & math.abs(
    bbox(1)-bbox(4))
184 end if
185 Next
186
187 if isValid then lw.WriteLine("Total Mass: " & Format(mMass, "0.000#") & "
    kg")
188
189 lw.WriteLine("
    *****")
190 lw.Close()
191 SavetoCSV()
192
193 End Sub
194
195 Private Sub SavetoCSV()
196 Dim FileName As String = FilePath & "\" & FnName & " - Mass.csv"
197
198 Dim stream_writer As New IO.StreamWriter(FileName, False)
199
200 Dim NameCut as string
201 if instr(FnName, "-") > 1 then
202 NameCut = FnName.Substring(0, FnName.IndexOf("-"))
203 else
204 NameCut = FnName
205 end if
206
207 stream_writer.Write("Upstream Box, Volume, Mass, Mass Ratio, Axial Length
    , Radial Length,
208 Z Axis Length, Downstream Box (External Id), Arrow Type" & vbCrLf)
209
210 Dim i As Integer
211 For i = 0 To UBound(CalcAttr)
212
213 stream_writer.Write(CalcAttr(i).Fname & ", " & Format(CalcAttr(i).FVolume
    , "0.000#")

```

```
214 & ", " & Format(CalcAttr(i).FMass, "0.000#") & ", " & Format((CalcAttr(i)
    ).FMass/mMass),
215 "0.000#") & ", " & Format(CalcAttr(i).AxialL, "0.0#") & ", " & Format(
    CalcAttr(i).RadialL,
216 "0.0#") & ", " & Format(CalcAttr(i).ZAxisL, "0.0#") & ", " & NameCut & "-
    Raw material cost"
217 & ", " & "Raw Material Contribution" & vbCrLf)
218 Next
219
220 stream_writer.Close()
221 End Sub
222
223 Public Function GetUnloadOption(ByVal dummy As String) As Integer
224
225 'Unloads the image immediately after execution within NX
226 GetUnloadOption = NXOpen.Session.LibraryUnloadOption.Immediately
227
228 End Function
229 End Module
```


Appendix G

GENERIC STEPS TO LOAD DATA INTO BOXARR

MODEL STRUCTURE

1. Import the WBS Structure.
2. Import Part Boxes.
3. Import Requirements (Need to manually pull the requirements structure into the CRD (WBS Element)).
4. Import Design Features.
5. Import Req to Feat Arrows.

FORGINGs

6. Import FORGING Value Streams WBS Elements (Untick the External Id, tick WBS Element (External Id) and change to External Id as the BOXARR field name).
7. Import FORGING Value Streams Boxes.
8. Import FORGING Value Streams Arrows (Change External Id to Downstream Box (External Id)).
9. Import FORGING Raw Material Contribution Arrows.

COS

10. Import COS Value Streams WBS Elements (Untick the External Id, tick WBS Element (External Id) and change to External Id as the BOXARR field name).
11. Import COS Value Streams Boxes.
12. Import COS Value Streams Arrows (Change External Id to Downstream Box (External Id)).
13. Import COS OPs WBS Elements.
14. Import COS Machining Data Boxes.
15. Import COS Machining Data Arrows (Change External Id to Downstream Box (External Id) and uncheck the Upstream Box (External Id)).

16. Import COS Machining Data Part Box Arrows (Change External Id to Downstream Box (External Id) and uncheck the Upstream Box and Arrow Type, tick the Part Box Arrow Type and select Arrow Type as the BOXARR field name).

FINISHED PART

17. Import Finished Part Value Streams WBS Elements (Untick the External Id, tick WBS Element (External Id) and change to External Id as the BOXARR field name).
18. Import Finished Part Value Streams Boxes.
19. Import Finished Part Value Streams Arrows (Change External Id to Downstream Box (External Id)).
20. Import Finished Part OPs WBS Elements.
21. Import Finished Part Machining Data Boxes.
22. Import Finished Part Machining Data Arrows (Change External Id to Downstream Box (External Id) and uncheck the Upstream Box (External Id)).
23. Import Finished Part Machining Data Part Box Arrows (Change External Id to Downstream Box (External Id) and uncheck the Upstream Box and Arrow Type, tick the Part Box Arrow Type and select Arrow Type as the BOXARR field name).

References

- Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56(6):568–585, 2014.
- Saeema Ahmed. Encouraging reuse of design knowledge: a method to index knowledge. *Design Studies*, 26(6):565–592, 2005.
- Gerardo Alducin-Quintero, Manuel Contero, Jorge Martin-Gutierrez, DavidA Guerra-Zubiaga, and MichaelD Johnson. Productivity improvement by using social-annotations about design intent in cad modelling process. In A. Ant Ozok and Panayiotis Zaphiris, editors, *Online Communities and Social Computing*, volume 6778 of *Lecture Notes in Computer Science*, pages 153–161. Springer Berlin Heidelberg, 2011.
- S. Ammar-Khodja and Alain Bernard. An overview on knowledge management. In Alain Bernard and Serge Tichkiewitch, editors, *Methods and Tools for Effective Knowledge Life-Cycle-Management*, pages 3–21. Springer Berlin Heidelberg, 2008.
- Najam Anjum, Jennifer A Harding, Robert IM Young, and Keith Case. Manufacturability verification through feature-based ontological product models. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2012.
- Stephane Aubry, Indira Thouvenin, Dominique Lenne, and Shigeki Okawa. Knowledge integration for annotating in virtual environments. *International Journal of Product Development*, 4(6):533–546, 2007.
- Marco Aurisicchio and Robert Bracewell. Capturing an integrated design information space with a diagram-based approach. *Journal of Engineering Design*, 24(6):397–428, 2013.
- Marco Aurisicchio, Rob Bracewell, and Ken Wallace. Understanding how the information requests of aerospace engineering designers influence information-seeking behaviour. *Journal of Engineering Design*, 21(6):707–730, 2010.
- Marco Aurisicchio, Rob Bracewell, and Gareth Armstrong. The function analysis diagram: Intended benefits and coexistence with other functional models. *AI EDAM*, 27 (Special Issue 03):249–257, 2013.

- Muhammad Imran Babar, Masitah Ghazali, Dayang N. A. Jawawi, Siti Maryam Shamsuddin, and Noraini Ibrahim. Phandler: An expert system for a scalable software requirements prioritization process. *Knowledge-Based Systems*, 84:179–202, 2015.
- Bojan Babic, Nenad Nesic, and Zoran Miljkovic. A review of automated feature recognition with rule-based pattern recognition. *Computers in Industry*, 59(4):321–337, 2008.
- Sebastian Barney, Aybuke Aurum, and Claes Wohlin. A product management challenge: Creating software product value through requirements selection. *Journal of Systems Architecture*, 54(6):576–593, 2008.
- Ira David Baxter. *Transformational maintenance by reuse of design histories*. PhD thesis, University of California at Irvine, 1991.
- Patrik Berander. *Evolving prioritization for software product management*. Doctoral dissertation, Deptment of Systems and Software Engineering, School of Engineering, Blekinge Institute of Technology, 2007.
- Joseph Bergin. Learning the planning game an extreme exercise. <http://csis.pace.edu/~bergin/xp/planninggame.html>, 2001. [Online; accessed 18-July-2016].
- Rahul A. Bidkar and Daniel A. McAdams. Feature recognition for injection-molded and die-cast parts. In *2004 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, September 28, 2004 - October 2, 2004*, volume 3 of *Proceedings of the ASME Design Engineering Technical Conference*, pages 693–702, Salt Lake City, UT, United states, 2004. American Society of Mechanical Engineers.
- Elke Bouillon, Patrick Mader, and Ilka Philippow. A survey on usage scenarios for requirements traceability in practice. In Joerg Doerr and Andreas L. Opdahl, editors, *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings*, pages 158–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- Jean-Francois Boujut and Julie Dugdale. Design of a 3d annotation tool for supporting evaluation activities in engineering design. In *Proceedings of the 7th International Conference on the Design of Cooperative Systems, COOP*, 2006.
- BOXARR. BOXARR Application. <http://www.boxarr.com/#!about-us/r3s62>, 2016. [Online; accessed 18-June-2016].
- Rob Bracewell, Ken Wallace, Michael Moss, and David Knott. Capturing design rationale. *Computer-Aided Design*, 41(3):173–186, 2009.
- Willem F. Bronsvoort, Rafael Bidarra, and Paulos J. Nyirenda. Developments in feature modelling. *COMPUTER AIDED DESIGN AND APPLICATIONS*, 3(5):655–664, 2006.

- Simon Buckingham Shum, Albert M. Selvin, Maarten Sierhuis, Jeffrey Conklin, Charles B. Haley, and Bashar Nuseibeh. Hypermedia support for argumentation-based rationale: 15 years on from gibis and qoc. In A. Dutoit, R. McCall, I. Mistrik, and B. Paech, editors, *Rationale Management in Software Engineering*, pages 111–132. Springer-Verlag, Berlin, 2006.
- Janet E. Burge. Design rationale: Researching under uncertainty. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(Special Issue 04):311–324, 2008.
- Jean David Caprace and Philippe Rigo. Towards a short time “feature-based costing” for ship design. *Journal of Marine Science and Technology*, 17(2):216–230, 2012.
- Par Carlshamre, Kristian Sandahl, Mikael Lindvall, Bjorn Regnell, and Johan Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 84–91, 2001.
- Neal Carter and Hayley Saunders. Trent Commodity Requirements Document (CRD)XWB-1000 HP Turbine Blade Commodity Requirements Document. Technical report, Rolls-Royce plc, 2015.
- Amareesh Chakrabarti, Stefan Morgenstern, and Helge Knaab. Identification and application of requirements and their impact on the design process: a protocol study. *Research in Engineering Design*, 15(1):22–39, 2004.
- Chiu-Cheung Chan. *Capturing design rationale within a CAD environment*. PhD thesis, Dept. of Industrial and Systems Engineering, The Hong Kong Polytechnic University, 2007.
- Senthil K. Chandrasegaran, Karthik Ramani, Ram D. Sriram, Imré Horváth, Alain Bernard, Ramy F. Harik, and Wei Gao. The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design*, 45(2): 204–228, 2013.
- Panagiota Chatzipetrou, Lefteris Angelis, Per Rovegard, and Claes Wohlin. Prioritization of issues and requirements by cumulative voting: A compositional data analysis framework. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 361–370, 2010.
- Aihua Chen. *A computer-based design history tool*. PhD thesis, Oregon State University, 1991.
- Gang Chen, Yongsheng S. Ma, Georg Thimm, and S. H. Tang. Unified feature modeling scheme for the integration of cad and cax. *Computer-Aided Design and Applications*, 1(1-4):595–601, 2004.

- Zheng-Ming Chen, Kun-Jin He, and Jing Liu. Automatic narrow-deep feature recognition for mould manufacturing. *Journal of Computer Science and Technology*, 26(3): 528, 2011.
- Jeff Conklin and Michael L. Begeman. gIBIS: A Hypertext Tool for Team Design Deliberation. In *Proceedings of the ACM Conference on Hypertext*, HYPERTEXT '87, pages 247–251, New York, NY, USA, 1987. ACM.
- Jeff Conklin and Michael L. Begeman. gibis: A hypertext tool for exploratory policy discussion. In *Proceedings of the 1988 ACM Conference on Computer-supported Cooperative Work*, CSCW '88, pages 140–152, New York, NY, USA, 1988. ACM.
- Jeffrey E. Conklin and Burgess K. C. Yakemovic. A process-oriented approach to design rationale. *Human-Computer Interaction*, 6(3):357–391, 1991.
- Alastair P. Conway, Matt D. Giess, Andrew Lynn, Lian Ding, Yee May Goh, Chris A. McMahon, and William J. Ion. Holistic engineering design: A combined synchronous and asynchronous approach. In *ASME 2008 International Design Engineering Technical Conference, Computers and Information in Engineering Conference (IDETC 2008)*, 2008.
- Stephen Cooper, Ip-shing Fan, Guihua Li, Britain Great, Trade Department of, Industry, University Cranfield, and Integration Department of Enterprise. *Achieving competitive advantage through knowledge-based engineering : a best practice guide*. Dept. of Enterprise Integration, Cranfield University, Cranfield [England], 1999.
- Andrew W. Court, Stephen J. Culley, and Chris A. McMahon. The influence of information technology in new product development: Observations of an empirical study of the access of engineering design information. *International Journal of Information Management*, 17(5):359–375, 1997.
- Nigel Cross. *Designerly ways of knowing*. Springer, London, 2006.
- Asa G. Dahlstedt and Anne Persson. Requirements interdependencies - moulding the state of research into a research agenda. In *Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2003), held in conjunction with CAiSE 2003*, pages 71–80, 2003.
- Asa G. Dahlstedt and Anne Persson. Requirements interdependencies: State of the art and future challenges. In Aybüke Aurum and Claes Wohlin, editors, *Engineering and Managing Software Requirements*, pages 95–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- Daniel Davies. *Representation of multiple engineering viewpoints in computer aided design through computer-interpretable descriptive markup*. Doctoral dissertation, University of Bath, 2008.

- Jesus M. de la Garza and S. Ramakrishnan. A tool for designers to record design rationale of a constructed project. In *10th International Conference on Applications of Artificial Intelligence in Engineering*, Southampton, UK, 1995.
- Peter Demian and Renate Fruchter. Effective visualisation of design versions: visual storytelling for design reuse. *Research in Engineering Design*, 19(4):193–204, 2009.
- Paolo Di Stefano, Francesco Bianconi, and Luca Di Angelo. An approach for feature semantics recognition in geometric models. *Computer-Aided Design*, 36(10):993–1009, 2004.
- Allen H. Dutoit, Raymond McCall, Ivan Mistrik, and Barbara Paech. Rationale management in software engineering: Concepts and techniques. In Allen H. Dutoit, Raymond McCall, Ivan Mistrik, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 1–48. Springer Berlin Heidelberg, 2006.
- Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requirements Engineering*, 14(4): 231–245, 2009.
- Gerhard Fischer, Raymond McCall, and Anders Mørch. Janus: Integrating hypertext with a knowledge-based design environment. In *Proceedings of the second annual ACM conference on Hypertext*, Pittsburgh, Pennsylvania, USA, 1989.
- Gerhard Fischer, Raymond McCall, and Anders Mørch. Design environments for constructive and argumentative design. *Special Interest Group on Computer and Human Interaction, ACM SIGCHI Bull.*, 20(SI):269–275, 1989.
- Pierre Foussier. *From product description to cost : a practical approach*. Springer, London, 2006.
- Jian Gao, Detao Zheng, and Nabil Gindy. Mathematical representation of feature conversion for cad/cam system integration. *Robotics and Computer-Integrated Manufacturing*, 20(5):457–467, 2004.
- Jian Gao, Detao T. Zheng, and Nabil Gindy. Extraction of machining features for cad/-cam integration. *The International Journal of Advanced Manufacturing Technology*, 24(7):573–581, 2004.
- Ana Cristina Garcia and Howard H. Craig Bicharra. Acquiring design knowledge through design decision justification. *AIEDAM Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 6(01):59–71, 1992.
- Sylvie Guibert, Francoise Darses, and Jean-Francois Boujut. Using annotations in a collective and face-to-face design situation. In Ina Wagner, Hilda Tellioglu, Ellen Balka,

- Carla Simone, and Luigina Ciolfi, editors, *ECSCW 2009*, pages 191–206. Springer London, 2009.
- Chun-Ling Hao. Design and implementation of nc lathe integrating cad system based on feature recognition. In *Intelligent Computation Technology and Automation (ICTA), 2014 7th International Conference on*, pages 162–166, 2014.
- Peter Heisig, Nicholas H. M. Caldwell, Khadidja Grebici, and P. John Clarkson. Exploring knowledge and information needs in engineering from the past and for the future – results from a survey. *Design Studies*, 31(5):499–532, 2010.
- Michael Himsolt. Graphlet: design and implementation of a graph editor. *Software Practice and Experience*, 30(11):1303–1324, 2000.
- Onur Hisarciklilar and Jean-Francois Boujut. An annotation based approach to support design communication. In *International Conference of Engineering Design, ICED’07*, Paris, France, 2007.
- Onur Hisarciklilar and Jean-Francois Boujut. An annotation model to reduce ambiguity in design communication. *Research in Engineering Design*, 20(3):171–184, 2009.
- Yoko Ishino and Yan Jin. Acquiring engineering knowledge from design processes. *AI EDAM*, 16(2):73–91, 2002.
- Timothy J. Jones, Carl Reidsema, and Adrian Smith. Automated feature recognition system for supporting conceptual engineering design. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 10(6):477–492, 2006.
- Wen-Ren Jong, Po-Jung Lai, and Tai-Chih Li. Integration and application of feature recognition and mould manufacturing planning navigating process. *International Journal of Production Research*, 52(20):5945–5964, 2014.
- Han JungHyun, Mike Pratt, and William C. Regli. Manufacturing feature recognition from solid models: a status report. *IEEE Transactions on Robotics and Automation*, 16(6):782–796, 2000.
- Peter Jurek, Bert Bras, Trine Guldberg, Jayne D’ Arcy, Seog-Chan Oh, and Stephan Biller. Activity-based costing applied to automotive manufacturing. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, 2012.
- Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.
- Joachim Karlsson, Stefan Olsson, and Kevin Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering*, 2(1):51–60, 1997.

- Joachim Karlsson, Claes Wohlin, and Bjorn Regnell. An evaluation of methods for prioritizing software requirements. *Information and Software Technology*, 39(14):939–947, 1998.
- Falak Khursheed and Mohammad Suaib. A survey on importance of requirement traceability in software engineering. *International Journal of Engineering and Technology Innovation*, 5(4):109–112, 2015.
- Sanghee Kim, Rob H. Bracewell, and Ken M. Wallace. Improving design reuse using context. In *International Conference of Engineering Design, ICED’07*, Paris, France, 2007.
- Mark Klein. Capturing geometry rationale for collaborative design. In *Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, 1997.
- Antje von Knethen and Mathias Grund. Quatrace: a tool environment for (semi-) automatic impact analysis based on traces. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*, pages 246–255, 2003.
- Antje von Knethen, Barbara Paech, Friedemann Kiedaisch, and Frank Houdek. Systematic requirements recycling through abstraction and traceability. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 273–281, 2002.
- Tom A. Kuffner and David G. Ullman. The information requests of mechanical design engineers. *Design Studies*, 12(1):42–50, 1991.
- Werner Kunz and Horst W. J. Rittel. *Issues as elements of information systems*. Institute of Urban and Regional Development, University of California, Berkeley, 1970.
- Gianfranco La Rocca. *Knowledge based engineering techniques to support aircraft design and optimization*. PhD thesis, Technical University of Delft, 2011.
- Antoon Hille van der Laan. *Knowledge based engineering support for aircraft component design*. PhD thesis, Technical University of Delft, 2008.
- Jintae Lee. Sibyl: a tool for managing group design rationale. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, Los Angeles, California, USA, 1990.
- Jintae Lee. Design rationale systems: understanding the issues. *IEEE Expert*, 12(3):78–85, 1997.
- Jintae Lee and Kum-Yew Lai. What’s in design rationale? *Human-Computer Interaction*, 6(3-4):251–280, 1991.

- Jintae Lee and Kum-Yew Lai. *A comparative analysis of design rationale representations*. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA., 1992.
- Dean Leffingwell and Don Widrig. *Managing software requirements: A Use Case Approach*. Addison-Wesley, Boston, MA, 2003.
- Laura Lehtola and Marjo Kauppinen. Empirical evaluation of two requirements prioritization methods in product development projects. In Torgeir Dingsøy, editor, *Software Process Improvement: 11th European Conference, EuroSPI 2004, Trondheim, Norway, November 10-12, 2004. Proceedings*, pages 161–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- Dominique Lenne, Indira Thouvenin, and Stephane Aubry. Supporting design with 3d-annotations in a collaborative virtual environment. *Research in Engineering Design*, 20(3):149–155, 2009.
- Chunley Li, Chris McMahon, and Linda Newnes. Annotation in design processes: Classification of approaches. In *International Conference of Engineering Design, ICED’09*, Stanford, CA, USA, 2009.
- Chunley Li, Chris McMahon, and Linda Newnes. Progress with ontocad: A standardised ontological annotation approach to cad systems. In *International Conference on Product Lifecycle Management (PLM11)*, Eindhoven, Netherlands, 2011.
- David J. Ling, Rajkumar Roy, Essam Shehab, Jay Jaiswal, and Janet Stretch. Modelling the cost of railway asset renewal projects using pairwise comparisons. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 220(4):331–346, 2006.
- Mitchell D. Lubars. Representing design dependencies in an issue-based style. *Software, IEEE*, 8(4):81–89, 1991.
- Allan MacLean, Richard M. Young, and Thomas P. Moran. Design rationale: The argument behind the artifact. *Conf Hum Fact Comput Syst Proc Conference on Human Factors in Computing Systems - Proceedings*, pages 247–252, 1989.
- Allan MacLean, Richard M. Young, Victoria M. E. Bellotti, and Thomas P. Moran. Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3-4):201–250, 1991.
- Allan MacLean, Victoria Bellotti, and Simon Shum. Developing the design space with design space analysis. In P. F. Byerley, P. J. Barnard, and J. May, editors, *Computers, Communication and Usability: Design issues, research and methods for integrated services*, volume North Holland Series in Telecommunication, pages 197–219. Elsevier, Amsterdam, 1993.

- Jatinder Madan, P. V. Madhusudhan Rao, and T. K. Kundra. Die-casting feature recognition for automated parting direction and parting line determination. *Journal of Computing and Information Science in Engineering*, 7(3):236–248, 2007.
- Anja M. Maier, P. John Clarkson, Matthias Kreimeyer, and Udo Lindemann. Reflecting communication: A key factor for successful collaboration between embodiment design and simulation. *J. Eng. Des. Journal of Engineering Design*, 20(3):265–287, 2009.
- Maksim Maksimovic, Ahmed Al-Ashaab, Essam Shehab, Myrna Flores, Paul Ewers, Badr Haque, Robert Furian, Frank von Lacroix, and Robert Sulowski. Industrial challenges in managing product development knowledge. *Knowledge-Based Systems*, 71(0):101–113, 2014.
- Raymond J. McCall. Issue-serve systems: A descriptive theory of design. *Design Methods and Theories*, 20(3):443–458, 1986.
- Raymond J. McCall. Phi: a conceptual foundation for design hypermedia. *Design Studies*, 12(1):30–41, 1991.
- Alison McKay, Saikat Kundu, Alan de Pennington, and Peter G. Dawson. An integrated product, process and rationale model for the provision of through-life information in product service systems. In *International Conference of Engineering Design, ICED'09*, Stanford, CA, USA, 2009.
- Chris McMahon, Alistair Lowe, and Steve Culley. Knowledge management in engineering design: personalization and codification. *Journal of Engineering Design*, 15(4):307–325, 2004.
- Haim Mendelson and Ravindran R. Pillai. Information age organizations, dynamics and performance. *Journal of Economic Behavior and Organization*, 38(3):253–281, 1999.
- Karen L. Myers, Nina B. Zumel, and Pablo Garcia. Acquiring design rationale automatically. *Artificial Intelligence for Engineering Design (AI EDAM)*, 14(02):115–135, 2000.
- Richard L. Nagy, David G. Ullman, and Thomas G. Dietterich. A data representation for collaborative mechanical design. *Research in Engineering Design Research in Engineering Design*, 3(4):233–242, 1992.
- Abouel E. Nasr, Awais A. Khan, Abdulrahman M. Alahmari, and Mohamed H. A. Hussein. A feature recognition system using geometric reasoning. *Procedia CIRP*, 18: 238–243, 2014.
- Andy Nolan, Olimpia Vlad, and Andrew C Pickard. The 10+/-2 factors for estimate success. Paper presented at ICEAA Symposium, Bristol, UK, October 2016.

- Patrick A. Oduguwa, Rajkumar Roy, and Peter J. Sackett. Cost impact analysis of requirement changes in the automotive industry: A case study. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 220(9):1509–1525, 2006.
- Roderick Owen and Imre Horvath. Towards product-related knowledge asset warehousing in enterprises. In *Proceedings of the Fourth International Symposium on Tools and Methods of Competitive Engineering, TMCE*, pages 155–170. HUST Press, 2002.
- Oxford. *Oxford Dictionary of English*. Oxford University Press, Oxford, 2010.
- Nicolas Perry and Samar Ammar-Khodja. A knowledge engineering method for new product development. *Journal of Decision Systems*, 19(1):117–133, 2010.
- Balasubramaniam Ramesh and Vasant Dhar. Supporting systems development by capturing deliberations during requirements engineering. *Software Engineering, IEEE Transactions on*, 18(6):498–510, 1992.
- Balasubramaniam Ramesh and Matthais Jarke. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, 2001.
- William C. Regli, Xiaobo Hu, Michael Atwood, and Wei Sun. A survey of design rationale systems: Approaches, representation, capture and retrieval. *Engineering with Computers*, 16(3-4):209–235, 2000.
- Bjorn Regnell, Barbara Paech, Aybuke Aurum, Claes Wohlin, Allen Dutoit, and Johan Natt Och Dag. Requirements mean decisions! - research issues for understanding and supporting decision-making in requirements engineering. In *Requirements Engineering”, Proc. 1st Swedish Conference on Software Engineering Research and Practice (SERP’01)*, pages 49–52, 2001.
- Jose Rios, Rajkumar Roy, and A. Lopez. Design requirements change and cost impact analysis in airplane structures. *International Journal of Production Economics*, 109(1-2):65–80, 2007.
- Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, 1973.
- William N. Robinson, Suzanne D. Pawlowski, and Vecheslav Volkov. Requirements interaction management. *ACM Comput. Surv.*, 35(2):132–190, 2003.
- Rolls-Royce. Rolls-Royce Capability Intranet. Technical report, internal presentation - ”Framework for reporting cost models for novel technologies”. Rolls-Royce plc [Internal Data Base], 2016.
- Rolls-Royce. Rolls-Royce Internal Data Base, 2016.

- Rajkumar Roy, Scott Colmer, and Terry Griggs. Estimating the cost of a new technology intensive automotive product: A case study approach. *International Journal of Production Economics*, 97(2):210–226, 2005.
- Christopher Rush and Rajkumar Roy. Expert judgement in cost estimating: Modelling the reasoning process. *Concurrent Engineering*, 9(4):271–284, 2001.
- Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, 1980.
- Phillip Sainter, Keith Oldham, Andrew Larkin, Adrian Murton, and Richard Brimble. Product knowledge management within knowledge-based engineering systems. In *Proceedings of DETC'00 ASME 2000 Design Engineering Technical Conference And Computers and Information in Engineering Conference*, Baltimore, Maryland, 2000. ASME.
- Albert Selvin, Simon Buckingham Shum, Maarten Seirhuis, Jeff Conklin, Beatrix Zimmerman, Charles Palus, Wilfred Drath, David Horth, John Domingue, Enrico Motta, and Gangminn Li. Compendium: making meetings into knowledge events. In *Knowledge Technologies 2001*, Texas, USA, 2001.
- Albert M. Selvin and Simon J. Buckingham Shum. Rapid knowledge construction: a case study in corporate contingency planning using collaborative hypermedia. *Knowledge and Process Management*, 9(2):119–128, 2002.
- Jami J. Shah and Martti Mantyla. *Parametric and feature-based CAD/CAM : concepts, techniques, and applications*. Wiley, New York, 1995.
- Randy H. Shih. *Parametric modeling with NX9*. SDC Publications, Mission, KS, 2014.
- Frank M. Shipmann and Raymond J. McCall. Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 11(2):141–154, 1997.
- Miroslav Skoda, Gabriela Slavikova, and Daniel Lajcin. The only correct calculation method in cost management: From activity based costing perspective. *Journal of Business and Management*, 3(3):8–18, 2014.
- Ram D. Sriram. *Intelligent systems for engineering : a knowledge-based approach*. Springer, London; New York, 1997.
- Simon Szykman, Ram D. Sriram, and William C. Regli. The role of knowledge in next-generation product development systems. *ASME Journal of Computing and Information Science in Engineering*, 1(1):3–11, 2001.

- Christopher Tong and Duvvuru Sriram. *Artificial intelligence in engineering design, volume I : design representation and models of routine design*. Academic Press, Boston, 1992.
- Adam Trendowicz. What is a good estimate? In Adam Trendowicz, editor, *Software Cost Estimation, Benchmarking, and Risk Assessment: The Software Decision-Makers' Guide to Predictable Software Development*, pages 9–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- David G. Ullman. Toward the ideal mechanical engineering design support system. *Research in Engineering Design*, 13(2):55–64, 2002.
- Jeroen Van Schaik. *A framework for design rationale capture and use during geometry design*. PhD thesis, University of Southampton, 2013.
- Vanguard. Vanguard Software Corporation. <http://www.vanguardsw.com/vanguard-studio/>, 2016. [Online; accessed 18-June-2016].
- Wim J. C. Verhagen, Pablo Bermell-Garcia, Reinier E. C. van Dijk, and Richard Curran. A critical review of knowledge-based engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1):5–15, 2012.
- David D. Walden, Garry J. Roedler, Kevin Forsberg, R. Douglas Hamelin, Thomas M. Shortell, and Engineering International Council on Systems. *Systems engineering handbook : a guide for system life cycle processes and activities*. 2015.
- Silvan Wiegeraad. *Development of a design history information system : capturing and re-using the knowledge behind the product*. Doctoral thesis, Technische Universiteit Eindhoven, 1999.
- Karl Eugene Wiegers. *Software Requirements*. Microsoft Press, 2003.
- Roland Wiese, Markus Eiglsperger, and Michael Kaufmann. yfiles: Visualization and automatic layout of graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers*, pages 453–454. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- Stefan Wiesner, Margherita Peruzzini, Jannicke Baalsrud Hauge, and Klaus-Dieter Thoben. Requirements engineering. In Josip Stjepandić, Nel Wognum, and Wim J.C. Verhagen, editors, *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*, pages 103–132. Springer International Publishing, Cham, 2015.
- Stefan Winkler and Jens von Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling*, 9(4):529–565, 2010.

- K. C. Burgess Yakemovic and E. Jeffery Conklin. Report on a development project use of an issue-based information system. In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*, CSCW '90, pages 105–118, New York, NY, USA, 1990. ACM.
- yWorks. yFiles for Java 2.x. <https://www.yworks.com/products/yfiles-for-java-2.x>, 2016. [Online; accessed 18-June-2016].
- Pei Zhan, Uma Jayaram, OkJoon Kim, and Lijuan Zhu. Knowledge representation and ontology mapping methods for product data in engineering applications. *Journal of Computing and Information Science in Engineering*, 10(2), 2010.
- Yingzhong Zhang and Xiaofang Luo. Design intent information exchange of feature-based cad models. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 3, pages 11–15, 2009.
- Yingzhong Zhang, Xiaofang Luo, Baiyun Zhang, and Shaohua Zhang. Semantic approach to the automatic recognition of machining features. *The International Journal of Advanced Manufacturing Technology*, pages 1–21, 2016.