CRANFIELD UNIVERSITY

Carlos A. Rodríguez-Toro

Product Complexity Assessment for a Proactive-DFA Implementation

(Simplicity + Simplicity ≅ Complexity)

SCHOOL OF INDUSTRIAL AND MANUFACTURING SCIENCE

PhD Thesis

CRANFIELD UNIVERSITY
SCHOOL OF INDUSTRIAL AND MANUFACTURING SCIENCE

PhD Thesis
Academic Year 2003 - 2004

Carlos A. Rodríguez-Toro

Product Complexity Assessment for a Proactive-DFA Implementation

(Simplicity + Simplicity ≅ Complexity)

Supervisor: Graham E.M. Jared

October 2004

This thesis is submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

ABSTRACT

This thesis presents product complexity as a criterion for the optimisation of product design in the light of an Assembly-Oriented Design and Design for Assembly implementation. It takes a holistic approach to the evaluation of the product architecture by presenting a set of indicators that help examine the product structure at two different levels: Assembly and Component complexity. Assembly complexity assessment is further sub-divided into Structural and Sequence complexity. The latter is a well-known and thoroughly studied area in assembly sequence evaluation, whereas the former gives a novel and original approach to drawing attention to those areas in the product configuration that will consume more resources (i.e. time and tooling required). Component complexity, on the other hand, is sub-divided into manufacturing and process handling/manipulation complexity. The first area has been addressed by the manufacturing analysis section of most Design for Assembly and Manufacturing methodologies, but it has been traditionally addressed as a manual and chart-based evaluation. This is a rigid approach that leaves little room for expansion and has no connection with the product structure. The metrics presented in this work embody a new approach that takes into account the component-to-component interactions and allows the analysis of component shape by extracting its geometry characteristics and comparing them with particular traits of the manufacturing processes available to the designer.

Additionally, the metrics presented in this work can be used to make an assessment of the product complexity at a particular point (static complexity) in the development cycle. They can also be registered over a period of time to provide an estimate of the possible consequences of the decisions made during a part of the development cycle (dynamic complexity). By using the methods developed, designers could reduce production costs and increase the reliability of their products.

ACKNOWLEDGEMENTS

Writing a thesis is a long process. Even with the proper supervision, it is a hard undertaking, it not only teaches the author about the subject in question, but about endurance, discipline and toughening of character. Over the past three years many people have helped me, not only in terms of the subject, with constructive criticism but with moral support. In particular my supervisor Graham Jared who has not only provided me with the necessary advice on what to put in the dissertation and what not, but more importantly with good-heartfelt advice as a human being and in the most needed circumstances. Throughout these years I have also received unconditional support from my parents (Rafael & Luz Marina), brother and sister (Juan Camilo & Elizabeth). They have looked after me even from afar and have constantly encouraged me to carry on during the toughest days. I have also met fantastic people that have supported me and taught me a lot about academic life and about life in general. Therefore I would like to thank Susan Tate (for her support since I came to the UK), Xana Lima and Angelica Oliveira (not only the best of friends, but both of them always made me feel at home amongst strangers and for that I love them dearly), Johnny Rodrigues, Mary Breen and Gabrielita their daughter (they made me feel part of their family), and my friends back home, who always had their welcoming arms open for me (Freire Diaz, Yamile Henao and with lots of love Luz Madelén Ramírez).

Finally I would like to acknowledge the support of Ken Swift who provided me with good ideas from a quick thinker.

CONTENTS

1 INTRODUCTION 1

2 THE DESIGNERS' SANDPIT PROJECT BACKGROUND 9

2.1 MOTIVATION 9

2.2 BACKGROUND 10

2.3 MAIN AREAS OF RESEARCH 11

2.3.1 Assembly-Oriented Design *12*

2.3.2 Traditional DFA implementation *12*

2.3.3 Proactive DFA implementation *14*

2.3.4 Product complexity analysis (an additional area of research) *15*

2.4 PRODUCT COMPLEXITY ANALYSIS 16

3 COMPLEXITY AND COMPLEX SYSTEMS: DEFINITIONS 19

3.1 DEFINITIONS OF 'COMPLEXITY' 19

3.2 COMPLEXITY TERMINOLOGY AND COMPLEX SYSTEMS 22

3.3 INFORMATION THEORY AND MANUFACTURING SYSTEMS 25

3.3.1 Tools and theories used in complex manufacturing systems *26*

3.3.2 Static complexity *31*

3.3.3 Dynamic complexity *32*

3.4 MODELLING AND LEARNING IN COMPLEX SYSTEMS 34

3.5 SCHOOLS OF THOUGHT 38

4 COMPLEXITIES OF PRODUCT DEVELOPMENT 41

4.1 COMPONENT-BASED ANALYSIS 43

4.1.1 Geometry *43*

4.1.2 Topology *44*

4.1.3 Shape complexity in support of DFX methodologies *45*

4.2 ASSEMBLY-RELATED ANALYSIS 46

4.3 SPECIFIC COMPLEXITY METRICS 46

4.3.1 Design complexity metrics *47*

4.3.2 Complexity metrics for DFX methods *51*

4.4 COMPLEXITY IN THE DESIGNERS' SANDPIT ENVIRONMENT 57

4.4.1 Role of Complexity Analyses *58*

4.4.2 Assembly Complexity *60*

4.4.3 Component Complexity *62*

4.5 FURTHER ISSUES FOR PRODUCT COMPLEXITY ASSESSMENT 62

4.5.1 Functional complexity: Design for Functionality *64*

4.5.2 Functionality interlink with structural decomposition ... 66
4.5.3 Product modularisation and variants: Design for Modularity ... 67
4.5.4 Product variety, change and flexible manufacture: Design for Variety ... 71

5 PRODUCT COMPLEXITY ASSESSMENT ... 75

5.1 COMPLEX SYSTEMS VS. COMPLEX PRODUCTS ... 75
5.2 PRODUCT METRICS ... 77
5.3 COMPLEXITY ACROSS SEVERAL STAGES OF DEVELOPMENT ... 78
5.4 KNOCK-ON EFFECTS IN PRODUCT DEVELOPMENT CYCLE ... 79
5.4.1 Non-linearity of complexity layers in the process ... 81
5.4.2 Trade-offs amongst several DFX methods ... 85
5.4.3 Traceability and dynamic planning ... 86
5.5 COMPLEXITY ASSESSMENT – RANKING THEORY ... 90
5.5.1 Spatial and functional complexity ... 92
5.5.2 Setting boundaries ... 93
5.6 COMPLEXITY TAXONOMY TOWARDS PROACTIVE DFA: (LEVELS OF ABSTRACTION) ... 94
5.6.1 Sources of complexity during concept and detailed design ... 95
5.6.2 Complexity Indicators – Conditions and diagnostics ... 97
5.6.3 Complexity indicators – product signature ... 99

6 ASSEMBLY COMPLEXITY ASSESSMENT ... 101

6.1 STRUCTURAL COMPLEXITY (PAIR-WISE INTERACTIONS) ... 102
6.1.1 Average Structural Complexity Index for Interactions ... 104
6.1.2 Component connectivity information in an assembly ... 106
6.1.2.1 Level of connectivity and deviation from minimum level ... 106
6.1.2.2 Boundaries for amount of interactions and variables held ... 109
6.1.2.3 Aiming for the boundaries ... 110
6.1.2.4 Upper bound: Theoretical and Real ... 113
6.1.2.5 Variables needed to track interactions ... 114
6.1.3 Component Interfaces in an Assembly (Connection types) ... 115
6.1.3.1 Kinematic interactions – geometric information ... 116
6.1.3.2 Geometric interactions ... 118
6.1.3.3 Adjacent parts (Contact) ... 120
6.1.3.4 Indirect interaction ... 122
6.1.3.5 Indirect interaction made irrelevant - Discussion ... 123
6.1.3.6 Non-exclusive component interactions ... 124
6.1.4 Complexity of individual interfaces ... 124
6.1.5 Matrix Arrangement of Interactions and Interfaces ... 127
6.1.5.1 Components interacting in a multi-dimensional space ... 129
6.1.5.2 Amount of information in a matrix-based representation ... 132
6.1.5.3 Component clusters ... 132

6.1.6 Periodicity – Subassemblies, arrays and pattern formation. 133
6.1.6.1 Component types 134
6.1.6.2 Standardisation 136
6.1.6.3 Re-usable information 136
6.2 CASE STUDIES ON CONNECTIVITY AND INTERACTION 137
6.3 STRUCTURAL COMPLEXITY – DYNAMIC 138
6.3.1 Sequence complexity 138
7 CASE STUDIES: ADJACENCY MATRIX EVALUATION (RESULTS) 141
7.1 PRESSURE RELIEF VALVE 144
7.1.1 Original design of the pressure relief valve 144
7.1.2 Matrix representation and graph representation 144
7.1.3 Complexity evaluation 145
7.1.4 Overall interface distribution graph 145
7.1.5 Comments 145
7.1.6 Suggested redesign of the pressure relief valve 146
7.1.6.1 Matrix and graph representation 147
7.1.6.2 Complexity evaluation 147
7.1.6.3 Overall interface distribution graph 148
7.1.6.4 Comments 148
7.2 OIL PUMP 149
7.2.1 Original design 149
7.2.1.1 Matrix and graph representation 150
7.2.1.2 Complexity evaluation 150
7.2.1.3 Overall interface distribution graph 151
7.2.1.4 Comments 151
7.2.2 Oil pump – suggested redesign 153
7.2.2.1 Matrix and graph representation 154
7.2.2.2 Complexity evaluation 154
7.2.2.3 Overall interface distribution 155
7.2.2.4 Comments 155
7.3 STAPLE REMOVER 157
7.3.1 Original design 157
7.3.1.1 Matrix and graph representation 157
7.3.1.2 Complexity evaluation 158
7.3.1.3 Overall interface distribution 158
7.3.1.4 Comments 158
7.3.2 Staple remover – suggested redesign 160
7.3.2.1 Comments 160
7.4 MOTOR DRIVE ASSEMBLY 161
7.4.1 Original design 161

7.4.1.1 Matrix and graph representation 162
7.4.1.2 Complexity evaluation 162
7.4.1.3 Overall interface distribution 163
7.4.1.4 Comments 163
7.4.2 Motor drive – suggested redesign 165
7.4.2.1 Matrix and graph representation 165
7.4.2.2 Complexity evaluation 165
7.4.2.3 Overall interface distribution 166
7.4.2.4 Comments 166
7.5 WIPER MOTOR 167
7.5.1 Original design 168
7.5.1.1 Matrix and graph representation 168
7.5.1.2 Complexity evaluation 169
7.5.1.3 Overall interface distribution 170
7.5.1.4 Matrix representation – reshuffled banded matrix 170
7.5.1.5 Comments 170
7.5.2 Wiper motor – suggested redesign 171
7.5.2.1 Matrix and graph representation 172
7.5.2.2 Complexity evaluation 172
7.5.2.3 Overall interface distribution 172
7.5.2.4 Comments 173
7.6 DOOR LATCH MECHANISM 174
7.6.1 Original design 174
7.6.1.1 Matrix and graph representation 175
7.6.1.2 Complexity evaluation 175
7.6.1.3 Overall interface distribution 176
7.6.1.4 Comments 176
7.6.1.5 Matrix representation – reshuffled banded matrix 177
7.6.2 Door latch – suggested redesign 178
7.6.2.1 Matrix and graph representation 178
7.6.2.2 Complexity evaluation 179
7.6.2.3 Overall interface distribution 180
7.6.2.4 Comments 180
7.7 RESULTS SUMMARY 181

8 COMPONENT COMPLEXITY ASSESSMENT 185

8.1 COMPONENT COMPLEXITY CLASSIFICATION 187
8.2 MANUFACTURING COMPLEXITY 188
8.2.1 Geometric information and shape complexity 188
8.2.2 Shape complexity – definitions 191
8.2.2.1 Shape Complexity Quantification – Initial approach 193
8.2.2.2 Loop-based quantification of shapes 194

8.2.2.3 Shape complexity and manufacturing process selection 197

8.2.3 Manufacturing process characterisation 197

8.2.3.1 Suggested analysis by characterised manufacturing processes 200

8.2.3.2 Foundations for a manufacturing process characterisation 205

8.2.3.3 Process characteristics – example 207

8.3 PROCESS HANDLING / MANIPULATION COMPLEXITY 209

9 FINAL DISCUSSION AND FUTURE WORK 211

9.1 COMPLEXITY ESTIMATION ACROSS MULTIPLE PRODUCT DOMAINS 212

9.2 PRODUCT DOMAINS: DEFINITIONS AND DIFFERENTIATIONS 214

9.3 COMPLEXITY MANAGEMENT AND MANUFACTURING BARRIERS 215

9.4 RELATIONSHIP BETWEEN PRODUCT DOMAINS (GENERAL VALUES) AND MANUFACTURING BARRIERS (SPECIFIC VALUES) 217

9.5 COMPLEXITY METRICS AND DECISION SUPPORT INTEGRATION 219

10 CONCLUDING REMARKS 223

10.1 THE IMPORTANT OF PRODUCT COMPLEXITY ASSESSMENT 225

10.2 INFORMATION EXTRACTED FROM THE CASE STUDIES 225

10.3 LESSONS LEARNED FROM THE COMPLEXITY METRICS STUDY 227

11 REFERENCES 230

12 APPENDIX A – ADJACENCY MATRIX 244

12.1 MATRIX REPRESENTATION 244

12.2 ADJACENCY MATRIX – ENTRY METHOD 245

13 APPENDIX B – SHAPE COMPLEXITY EVALUATION 250

LIST OF FIGURES

Figure 2.1 – (a) Bottom-Up vs. (b) Assembly-Oriented Design.........12
Figure 2.2 – Implementation of Proactive DFA (adapted from Tate et al. [5]).........14
Figure 3.1 – Complex systems framework and manufacturing issues (adapted from McCarthy et al [19]).........26
Figure 3.2 – System responsiveness in the edge of chaos.........27
Figure 4.1 – Hinckley's Assembly defect vs. Complexity Factor.........53
Figure 4.2 – Reverse Fishbone Diagram (RFD) (After Lee et al., 1997).........54
Figure 4.3 – Two relation hierarchies with the same Boothroyd-Dewhurst's complexity factor value (After Balasz).........56
Figure 4.4 – Different complexity types considered at a particular level.........60
Figure 4.5 – Architectural and Functional Hierarchies (Function-to-part mapping).........64
Figure 4.6 – Modularisation.........68
Figure 5.1 – Influence of component complexity and part count in product cost.........77
Figure 5.2 – Design changes vs. Product development cost.........78
Figure 5.3 – Several sources of complexity across design stages.........80
Figure 5.4 – Development process pipeline.........80
Figure 5.5 – Layers of complexity added to the product as it follows the design process.........81
Figure 5.6 – Risks presented when adjusting the level of complexity at anyone stage.........82
Figure 5.7 – Histogram. Complexity at every time step for risk evaluation (consequences).........87
Figure 5.8 – Environmental constraints and feedback in product development.........94
Figure 5.9 – Complexity analysis defined in terms of Component vs. Assembly and Static vs. Dynamic.........95
Figure 5.10 – Complexity Indicators – Monitors.........98
Figure 5.11 – Product structures to be compared.........99
Figure 6.1 – Components interacting in an assembly.........102
Figure 6.2 – (a) Single interaction vs. (b) Various types of component interactions.........103
Figure 6.3 – Component interaction complexity.........106
Figure 6.4 – Simple chain of components.........106
Figure 6.5 – (Top) Lower and (Bottom) Upper bound interaction graph.........110
Figure 6.6 – Upper and Lower Bound for the number of interactions.........111
Figure 6.7 – Real Upper and Lower Bound - Modifications to the assembly.........114
Figure 6.8 – Graph for the number of variables held at one time.........114
Figure 6.9 – Six Lower Pairs (a) revolute or pin; (b) prism; (c) helical; (d) cylindric; (e) spheric; and (f) planar. (Adapted from Uicker Jr, Pennock and Shigley [160]).........117
Figure 6.10 – Geometric docking.........119
Figure 6.11 – Geometric coupling (third component required).........119
Figure 6.12 – Tribocontact (Friction & wear).........120

Figure 6.13 – Components in contact constraining each other *121*

Figure 6.14 – Elements in sporadic contact *122*

Figure 6.15 – Indirect component interaction *122*

Figure 6.16 – Indirect component interaction vs. Geometric interaction *123*

Figure 6.17 – Graph representation of interactions and interfaces *127*

Figure 6.18 – Matrix representation of component interactions in binary space *128*

Figure 6.19 – Component interactions in a multi-dimensional space *130*

Figure 6.20 – Information contained in a cell; (a) Component to component; (b) Bits of information per cell *132*

Figure 6.21 – (a) Original matrix representation of a product structure, (b) Re-shuffled matrix to detect component clusters. (Taken from case studies in Appendix B) *133*

Figure 6.22 – Complexity affected by arrangement periodicity *134*

Figure 6.23 – Number of various types of components in an assembly *135*

Figure 7.1 – Pressure relief valve (original design) *144*

Figure 7.2 – Matrix representation of pressure relief valve (original design) *144*

Figure 7.3 – Graph representation of pressure relief valve (original design) *144*

Figure 7.4 – Pressure relief valve (original design): Interface distribution graph *145*

Figure 7.5 – Pressure relief valve (suggested redesign) *147*

Figure 7.6 – Matrix representation for Pressure relief valve (suggested redesign) *147*

Figure 7.7 – Graph representation for pressure relief valve (suggested redesign) *147*

Figure 7.8 – Pressure valve (redesign) : Interface distribution graph *148*

Figure 7.9 – Oil pump (Original design) *149*

Figure 7.10 – Matrix representation for Oil Pump (original design) *150*

Figure 7.11 – Graph representation for Oil Pump (original design) *150*

Figure 7.12 – Oil pump (original design): Overall interface distribution graph *151*

Figure 7.13 – Oil pump (suggested redesign) *153*

Figure 7.14 – Matrix representation for Oil pump (suggested redesign) *154*

Figure 7.15 – Graph representation for Oil pump (suggested redesign) *154*

Figure 7.16 – Oil pump (redesign): Overall interface distribution *155*

Figure 7.17 – Staple remover (original design) *157*

Figure 7.18 – Matrix representation for Staple remover (original design) *157*

Figure 7.19 – Graph representation for Staple remover (original design) *157*

Figure 7.20 – Staple remover (original design): Overall interface distribution *158*

Figure 7.21 – Staple remover (suggested redesign) *160*

Figure 7.22 – Motor drive assembly (original assembly) *161*

Figure 7.23 – Matrix representation for Motor drive assembly (original design) *162*

Figure 7.24 – Graph representation for Motor drive assembly (original design) *162*

Figure 7.25 – Motor drive (original design): Overall interface distribution. *163*

Figure 7.26 – Motor drive assembly (suggested redesign) *165*

Figure 7.27 – Matrix representation for Motor drive assembly (suggested redesign). *165*
Figure 7.28 – Graph representation for motor drive assembly (suggested redesign) *165*
Figure 7.29 – Motor drive (suggested redesign): Overall interface distribution. *166*
Figure 7.30 – Wiper motor (Original design). *168*
Figure 7.31 – Matrix representation for wiper motor (original design) *168*
Figure 7.32 – Graph representation for wiper motor (original design). *169*
Figure 7.33 – Wiper motor (original design): Overall interface distribution. *170*
Figure 7.34 – Matrix representation for Wiper motor (original design) - Banded matrix. *170*
Figure 7.35 – Wiper motor suggested redesign. *171*
Figure 7.36 – Matrix representation for wiper motor (suggested redesign). *172*
Figure 7.37 – Graph representation for wiper motor (suggested redesign). *172*
Figure 7.38 – Wiper motor (suggested redesign): Overall interface distribution. *172*
Figure 7.39 – Door latch original design. *174*
Figure 7.40 – Matrix representation for Door latch (original design). *175*
Figure 7.41 – Graph representation for Door latch mechanism (original design). *175*
Figure 7.42 – Door latch (Original design): Overall interface distribution. *176*
Figure 7.43 – Matrix representation for Door latch (original design) - Banded matrix. *177*
Figure 7.44 – Door latch suggested redesign. *178*
Figure 7.45 – Matrix Representation for Door Latch (suggested redesign). *178*
Figure 7.46 – Graph representation for Door latch (suggested redesign) *179*
Figure 7.47 – Door latch (suggested design): Overall interface distribution. *180*
Figure 8.1 – Design Methodologies; (a) Bottom - Up vs (b) Top – Down design *186*
Figure 8.2 – Component Shape; (a) A collection of interfaces that determine the component's overall external shape; (b) Component shape example *187*
Figure 8.3 – Component complexity classification *188*
Figure 8.4 – Component classification based on overall shape *189*
Figure 8.5 – Axial 'Protrusions' in Rotational Components *193*
Figure 8.6 – Parent-Child Face Relationships *195*
Figure 8.7 – General classification of manufacturing processes (Adapted from Swift and Booker [184]) *199*
Figure 8.8 – General view of manufacturing analysis supported by process characterisation *200*
Figure 8.9– Manufacturability analysis supported by a process characterisation engine (detail) *201*
Figure 8.10 – Component Feature characterisation *202*
Figure 8.11 – Shell moulding process *208*
Figure 8.12 - Powder metallurgy process *209*
Figure 9.1 – Classification of 3D models: Geometric complexity versus Combinatorial complexity (taken from Shikhare [60]) *213*
Figure 9.2 – Product complexity across application domains *213*
Figure 9.3 – General overview of Complexity vs. Number of parts *216*

Figure 9.4 – Complexity metrics and decision support integration 219
Figure 10.1 – DFA-optimised product structure's evolution 226
Figure 12.1 – Graphs representations (a) lattice and (b) trees 245
Figure 12.2 – Part-to-part representation methods 245
Figure 12.3 – How to fill in the adjacency matrix. (a) Matrix formation and (b) entries. 246
Figure 12.4 – Weighted interactions - Interface types 247
Figure 12.5 – (a) Adjacency matrix for any product architecure, (b) Equivalente on a graph representation 247
Figure 13.1 – Part Envelope and wall categories 250
Figure 13.2 – Shape classification categories 251

LIST OF TABLES

Table 4.1 – Halstead's vs. Braha-Maimon's metrics......50
Table 4.2 – MacDuffie's Product complexity measures in product variety......71
Table 5.1 – General aspects considered for complexity evaluation......96
Table 6.1 – Evaluation of components interactions (example)......103
Table 6.2 – Components interaction - example......106
Table 6.3 – Number of variables held at one time for a product due to pair-wise interactions alone......114
Table 6.4 – Lower Pairs (From Uicker Jr. Pennock and Shigley)......117
Table 6.5 – Higher Pairs Joints......118
Table 6.6 – Nomenclature and numerical values for component interfaces......125
Table 6.7 - Contact/ Adjacency estimations for component interfaces.......126
Table 7.1 - Case studies result summary......182
Table 8.1 - Shell moulding process properties......208
Table 8.2 – Powder metallurgy process......209

Chapter 1

INTRODUCTION

Design optimisation is often seen as an iterative process. Ideas or concepts are continuously produced and turned into virtual or physical objects that are inspected time and time again until a refined solution is reached. Design optimisation is also regularly seen as an evolutionary process. Designers try to mimic the constant development process seen in designs created by nature. Nevertheless, designers do not have the luxury afforded to nature: time. Time is the only resource that is not in abundance (if available at all in engineers' schedules). Designers cannot spend more than a certain number of hours developing and perfecting a product. Time is on nature's side and its experiments (evolution) are performed in parallel to adapt to the environment and produce the most efficient design (solve a problem).

The term "adaptation" is therefore a key issue in evolution. To follow suit, designers have to adapt themselves to the ever-changing environment of business, although time is against them, the current computing landscape offers an increasing processing power, that can empower them by helping them achieve better and faster results, but it comes at a price. Resources (other than time itself) are small and limited Designers cannot compete against nature, but they can certainly learn from it. They are aware that nature wins when less is more, that is, nature ultimately manages to create "products" that perform all the required tasks with the minimum number of components. For nature efficiency is everything.

Designers have tried to imitate nature's designs [1], based on the fact that these designs have been tested over years of evolution and are considered the best possible solution to the problem they try to solve. However, rather than blindly copying what nature has perfected, scientists are willing to dissect these creations and understand their building blocks. Designers, nonetheless, have a major hindrance: their own mental images of solutions. The road from conception to production is populated with a myriad of factors that spawn millions of possibilities, ultimately rendering the challenge of design a daunting task. Fortunately, designers do not often see all these possibilities, but rather the end product as their goal. Sadly, because many chances for optimisation are often overlooked, there is a missed chance for understanding why these opportunities are the ones that may or may not decrease product manufacturing costs and make the difference between success and failure.

It is clear that the more variables need to be controlled, the more cumbersome the design process becomes. It is not surprising then that companies are becoming increasingly more interested in managing these variables, which convey the so-called "product complexity." Organisations, managers, engineers and academics are becoming more willing to understand the implications of the complexity of the product and to put in resources dedicated to its management. Industries today invest large amounts of money, time and other resources in acquisition and implementation of Computer-Aided Engineering (CAE) tools that comprise *Engineering Data Management* (EDM) (or *Product Data Management* – PDM, for that matter), as well as, Product Lifecycle Management (PLM) capabilities.

The evaluation and management of product complexity is therefore gaining momentum, and this can be seen in the currency achieved by terms such as "Design for Simplicity" or "Product Complexity Management", which are becoming progressively more popular. There are written reports about how companies are adopting already existing design optimisation tools, such as Design for Assembly (DFA), to boost their productivity and reduce costs. Just a handful of industries modify these tools to encourage simplicity in their designs [2]. Yet the fact that complex designs can still be produced and reasons as to why they become complex have not been clearly studied as such.

It follows that the notion of 'Complexity', whatever the field, has captured the imagination and interest of scientists and engineers for decades, and their attempts to gain an understanding can be seen in the various definitions given. As arguably a subjective characteristic, complexity in information theory and social systems has been associated with the overwhelming difficulty of, and subsequent effort required to understand such systems, thus inhibiting the prediction of their behaviour.

The concept of complexity poses an additional challenge, for its very definition is elusive and hard to grasp. Nevertheless, in order to offer an early indication of how complexity is understood in this thesis, the following can be taken as an initial working definition:

> *"Complexity is equivalent to the amount of resources and time invested in understanding and predicting the behaviour of a system."*

However, as the discussion in this thesis evolves, it will become more apparent that this definition fails to embrace all sources and factors that can contribute to the complexity of a system. For instance, this definition does not elaborate on whether the notion of complexity is purely subjective or in fact a property of the system being studied. The reason behind this last statement is the need to establish a 'quantitative measure of complexity' and therefore be able to 'estimate how complex a system would be'. Any estimation of complexity would, consequently, help to assess the amount and value of the effort invested in its understanding. If complexity is in fact a subjective property or rather a property of the observer, then, as a result, any sort of metric produced will be subjective, defeating the whole purpose of quantification. Subjective quantifications cannot be used for comparison, because they would be dependent on the user, and for that reason, an objective measure is required.

Assuming that complexity can be estimated and/or quantified, an additional question arises: should a single measure be produced? A single measure of complexity could be enough to produce a quick assessment of the system, but it would not be suitable for optimisation purposes. An overall estimation of complexity will expose the fact that such a system requires a large amount of resources, but it will not assist the user on how to make that system more efficient – same system performance, less amount of resources required. What is needed is 'a set of complexity metrics', a framework that provides the necessary support for optimisation. This framework can be achieved by introducing supervisory and monitoring controls (or indicators) to the system. Such a set of indicators will point out which areas are consuming much of the resources, therefore indicating where to direct most of the necessary intelligence and optimisation tools to improve the behaviour of the system. This system can be established with the evaluation of two different architectures of the same product, for instance, a parallel experiment to determine the best possible solution, or the same or different elements arranged in a variety of ways. The framework proposed in this thesis is a model that needs to be refined; nevertheless it still shows that its implementation is possible. The author is aware of the warning issued by Ulrich in terms of product structure evaluation: *"A single model of most of the trade-off associated with the choice of a product architecture is unlikely, and even if it were developed, it wouldprobably be too complex to be useful" (Ulrich, 1995[3])*

The summary presented above is the main intent of this research. However, there are a large number of points that need be addressed first. For instance, there are several types of systems (information systems, social systems, manufacturing systems amongst others). This work is exclusively related to manufacturing systems, where components are engineered entities stripped of any self awareness or behaviour of their own and, thus, avoiding any internal and unpredictable characteristics they might introduce[1]. Additionally, this work attempts to analyse the complexity existing in some of the stages of designing and developing a product. Considerations given to the term 'product' are the same as those expressed by Ulrich and Eppinger [4], where a 'product' is restricted to the notion of an engineered, physical and discrete entity, whereby a difference between a 'product' and a 'system' is stated.

Many authors regard a *product as a system*. This assumption is typically based on the presupposed similarities between the two concepts (e.g. number of components and interactions amongst components). Nevertheless, as its name implies, a *product is the outcome of a manufacturing system*, and therefore as mentioned above, it is discrete and closed as opposed to the open-ended characteristic of a system.

The analysis of complexity in the product development cycle is part of the goals established by The Designers' Sandpit research project [5, 6]. This project, described in Chapter 2, makes use of design evaluation tools already mentioned such

[1] In manufacturing systems, components are seen as "objects", a term borrowed from software terminology in object-oriented programming.

as Design for Assembly (DFA). These tools have demonstrated some success in reducing problems such as redesign, rework and a lengthy product introduction period - mainly resulting from traditional trends of using component-oriented CAD tools rather than dealing with a product as a whole. The main goal is to develop an environment for "Assembly-Oriented Design" incorporating methods for the generation and evaluation of concept design ideas, manufacturing analysis, assembly planning and design advice.

In terms of manufacturing processes, assembly costs and quality of the end product, complexity plays a vital role in the achievement of the best design that not only takes into account the assembly planning, but also the selection of the most suitable manufacturing process. One outcome of this work is the development of means to quantify the complexity of a product and the definition of measures to be used in conjunction with other metrics and enable comparison of shape similarities. One might wonder, nonetheless, what are complexity metrics good for? Based on this work, it can be said that they are good for giving instant feedback to the user, allowing him to have a picture of possible consequences of whatever decision s/he makes. On their own right, complexity metrics can help with the prediction of how factors modified in the product structure might behave and hence encourage adaptation.

However, exploring the scientific community for ideas and works one can see that, unfortunately, little has been achieved in the area of complexity metrics that can be used in a sensible way. One survey by Tang and Salminen [7] shows that from a series of studies, only 20% have attempted to produce some sort of quantification. Thus further research is required to make complexity a practically useful concept. This exploration for relevant works that lead to a definition of complexity and complex systems is presented in Chapter 3, paying special attention to the hierarchical characteristics of systems and in fact to the holistic approach to studying the notion of complexity assessment. Accordingly, it is followed by a report on state-of-the-art methodologies conceived to understand what complexity is in the specific area of product development, as presented in Chapter 4 (*Complexity in product development and design.*)

The rest of the thesis further develops those concepts, presenting novel ideas and possible ways to establish the above mentioned framework of complexity assessment. The main purpose, however, is the integration of complexity metrics with design evaluation tools such as DFA to create a proactive implementation of them, supporting the reasoning and decision making process designers use. More specifically, this work looks at the metrics of product architecture oriented towards an assembly-oriented design. Bearing this in mind, the rest of the thesis is organised as follows. Chapter 5 (*Product complexity assessment*) presents the different factors that contribute to the generation of complex designs, from an architectural point of view, but raises the question of its integration with evaluation of functional complexity. This sets the path to follow in terms of assembly and component complexity.

Consequently, Chapter 6 (*Assembly complexity assessment*) reports on assembly complexity. It explores assembly as an architectural/structural issue. This new exploration links product architecture with its assembly capabilities or difficulties, therefore lending itself to novel approach and convenient evaluation of its complexity. It is precisely in this chapter where estimations of complexity based on component binary interactions, the types and amount of interactions (referred to as *interfaces*) are presented. This chapter closes with suggestions on implementation of already existing metrics in the area of assembly sequence complexity. Accordingly, Chapter 7 presents a set of case studies based on DFA analysis of a series of products, where original and suggested redesigns are evaluated to learn how complexity has been tackled and whether it has been reduced or coped with.

In terms of component evaluation, Chapter 8 (*Component complexity assessment*) develops on estimations of component complexity, divided into shape complexity and component handling analysis. The latter is an essential part of the DFA analysis and is presented as such, whereas the estimation of shape complexity is further revised suggesting a novel approach, proposing an algorithm that interrogates every component based on manufacturing process characteristics.

It has been generally accepted that the study of complexity and tools to deal with it are part of an ongoing research process, for that reason, Chapter 9 (*Further* *discussion and future work*) indicates additional issues that need to be addressed to complement a suitable product-complexity evaluation framework.

Finally, Chapter 10 presents the conclusions that have been reached so far. These remarks summarise the knowledge gained throughout the research period and proposes different views on complexity issues, such as the evaluations across product domains and their integration with already existing metrics. To complement the work, Appendix A introduces one of methods used to assess architectural complexity, namely, an adjacency matrix. As a final point, Appendix B shows the existing charts used to estimate component shape complexity, which were revised in Chapter 8.

Chapter 2

THE DESIGNERS' SANDPIT PROJECT

Product complexity assessment and management is becoming of increasing importance to organisations and designers as a means of dealing with, controlling and monitoring the best use of manufacturing resources. This assessment has also captured their interest, in particular, for the possibilities that it offers in terms of implementation within CAE applications with Product Data Management capabilities.

In order to address these issues of integration and implementation, the Designer's Sandpit project is currently developing an environment that has, as one of its objectives, the effective implementation of these 'product-complexity-assessment' capabilities [6]. The project builds on previous work that sought to integrate DFA evaluation within a CAD environment [8, 9]. This previous effort made available the creation of an "assembly-oriented design environment", with emphasis on the development of techniques for optimum assembly sequence generation, the implementation of manual DFA evaluation methodologies and the development of geometric reasoning techniques to automatically extract data from CAD models. The Designers' Sandpit project seeks to extend these capabilities of DFA evaluation within a CAD environment, to further create an environment that supports designers in making the most effective decisions throughout the early stages of the product development process [5].

2.1 Motivation

In technological terms, the expression "sandpit" has recently become a synonym for an environment oriented to experimentation and learning. It is precisely through this idea of experimentation and learning that designers can develop and construct innovative products that implement best design practices, such as Design for Assembly, Design for Manufacture and, in general, all the techniques commonly known as DFX (Design for 'X'). Furthermore, the Designers' Sandpit seeks to address the creation of a much needed environment where designers are supported in their decision making process during the early stages of the product development cycle. It is now widely accepted that it is much more cost-effective to produce an initial design that is simple, feasible and assembly efficient, than it is to rectify design problems after the product has reached the shopfloor. Designers can dramatically benefit from a system that allows them to compare different ideas, in terms of product configurations, and therefore select the arrangement that best suits the product requirements. In doing so, designers can focus on product performance, that is, they can evaluate much more quickly how well the product achieves and delivers its required functionality.

The evaluation of product architecture can be performed with the implementation of both high and also detailed level evaluation metrics. These metrics can possibly be deduced from any geometric data that might be available at the early stages, but most certainly, they can be inferred from the architecture of the product. In effect, product configurations can start to be analysed and evaluated at a higher and more abstract level, before geometric information is available. This analysis is primarily based on how components interact in a pairwise manner. Clearly, the one key tool within such decision-support environment is the evaluation of product complexity at various stages of the product design.

2.2 Background

The Designers' Sandpit project is based on the idea of integrating CAD systems with DFA evaluation, as mentioned above. However, rather than developing this evaluation around traditional 'component-oriented design' systems, where parts are modelled then assembled to create the final product, the Designer's Sandpit goes a step further by providing an evaluation based on a top-down design methodology. This methodology can be implemented through an 'assembly-oriented design' environment [10]. The latter offers a holistic view of the product, where assembly issues can be addressed even before they cause problems downstream.

Additionally, the project seeks to implement a DFA analysis in an altogether different manner. Traditionally, DFA has been applied to complete products, thus requiring definitive geometry. However, this requirement hinders its efficacy, because once a product has been formalised, it is usually too late to make significant changes. The significant difference in the approach to integrating DFA with CAE systems between this project and traditional computer-based implementations is the concept of proactive DFA integration adopted by the Designers's Sandpit. This proactive DFA implementation in a CAE system requires the integration of: product functionality representation, product structure, and assembly sequence. The integration of these is necessary to make possible a meaningful analysis both at the early stages and throughout the development of the product design.

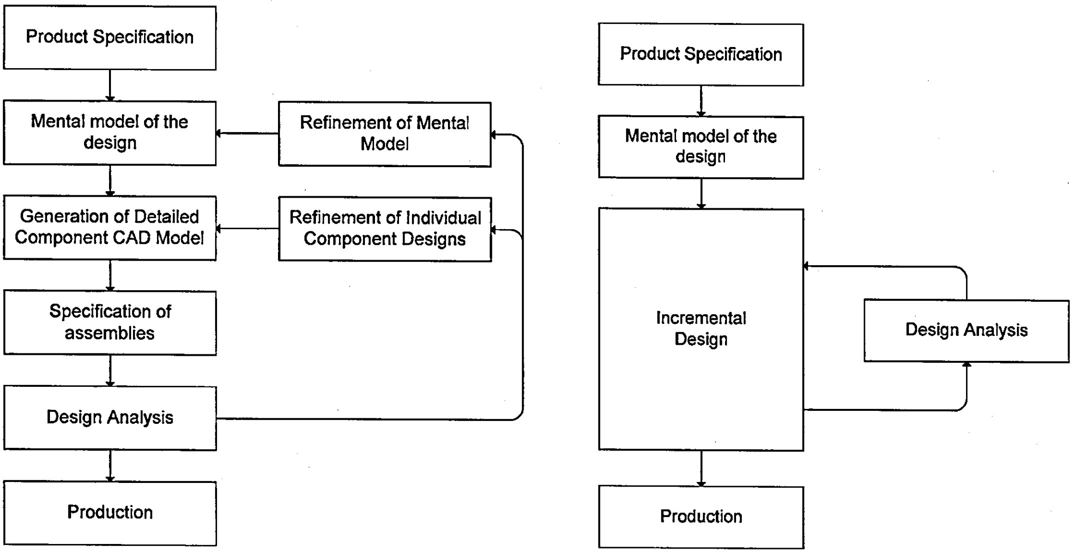
One section of the DFA methodology comprises an evaluation of the manufacturing processes required for each component of the product [11]. Such an evaluation is based on a subjective shape complexity analysis, similar to that used by part coding systems. However, this analysis is carried out as part of the inherent iterative process involved in any DFA implementation. Therefore, as parts are integrated or eliminated to optimise assembly with respect to part count and assembly operations, the manufacturability analysis might suggest that the newly created components could be too costly to be manufactured. This is one of the key trade offs during product design optimisation.

2.3 Main areas of research

The project seeks to explore and extend different areas related to the development of cost-efficient products. These areas of exploration are the creation of an assembly-oriented design environment, the implementation of Proactive DFA analysis in the earlier stages of the process development cycle, and the evaluation of product complexity to accomplish this proactive integration. These first two of these topics will be described briefly in the rest of this chapter.

2.3.1 Assembly-Oriented Design

Several authors have reported on the integration of a CAD environment with DFA as a significant step for consideration of assembly issues [12, 13]. Nevertheless, an 'assembly-oriented design' methodology provides the best platform for a top-down (see Figure 2.1) design environment that, if integrated with DFA analyses, can account for the construction, validation and evaluation of the assembly sequence and manipulation of the product structure.


Product Specification
Mental model of the design
Refinement of Mental Model
Generation of Detailed Component CAD Model
Refinement of Individual Component Designs
Specification of assemblies
Design Analysis
Production
Product Specification
Mental model of the design
Incremental Design
Design Analysis
Production

(a) (b)

Figure 2.1 – (a) Bottom-Up vs. (b) Assembly-Oriented Design

Furthermore, this assembly-oriented design environment provides the best approach to an integrated and cost effective product. Products can be evaluated for performance, considering the number of components required, even before geometric information is provided. Additionally, this holistic approach allows the comparison of different layouts of the same product, and thus being able to select the most cost effective of them.

2.3.2 Traditional DFA implementation

Product design optimisation methodologies such as DFA have been hailed as suitable approaches to assist designers to evaluate and thereby create the most efficient designs, from an assembly viewpoint. The different DFA techniques available follow the same format. The Designers' Sandpit uses an extended and improved version of the Lucas DFA methodology. This method operates at four different levels, namely:

- ***Functional Analysis*** – This determines the overall function of the product. It poses a series of minimum parts criteria to establish if each part is essential or non-essential, thus calculating a design efficiency value. The identification of every part and the design efficiency value of the product encourages the designer to optimise (or more accurately reduce) the product part count.
- ***Manufacturing Analysis*** – This allows designers to anticipate the component manufacturing costs associated with part count optimisation. It complements a traditional DFA analysis with the exploration of alternative materials and manufacturing technologies, enabling the assignment of relative cost levels to alternative component designs.
- ***Handling Analysis*** – It is concerned with the practicalities of manipulation and transport of components to their assembly points. Every component is awarded a score dependent on its ease of handling in terms of: size, weight, handling difficulties and ease or difficulty of orientation of the part, amongst others.
- ***Assembly Analysis*** – This final step is carried out after constructing an assembly sequence flowchart, which diagrammatically represents the order in which each part is assembled. This analysis captures the problems associated with the build sequence, assembly and test problems, and tooling requirements of the design.

Despite their benefits, DFA methodologies have traditionally been applied when a complete product description is available. This certainly undermines the purpose of product optimisation, as it is often seen as a reactive improvement, usually overlooked by most designers due to time constraints during the development of a product.

One attempted solution to this dilemma was through computer implementations of DFA analysis. However, these are mere translations of a manual approach to an automated operation, with little improvement other than speeding up the already late evaluation. A better option would seem to be that of taking this evaluation towards the earlier stages of the product development cycle, thus providing a much more appreciated and convenient support to designers.

2.3.3 Proactive DFA implementation

One of the main goals of the Designers' Sandpit project is the implementation of a decision-support system that helps designers achieve a 'right-first-time' design. It was previously stated that traditional implementations of DFA methodologies, although improving the configuration of the product considerably, must be considered as reactive tools, applied too late the development cycle (formalisation has already taken place) which are also heavily dependent on geometry information. Therefore, in order to improve the effectiveness of DFA a proactive, rather than a reactive, approach is required. This proactive implementation would certainly provide designers with the necessary tools for design evaluation when most needed: *during the earlier stages of the design and development process*.

Since a proactive DFA implementation would enable designers to identify necessary changes at the first available opportunity, then it is important to integrate the proposed proactive DFA analysis into the early stages where most of the relevant decisions take place. This stage must be when the product structure is outlined. In fact, the integration is proposed to take place in three levels of operation: Product Group, Product Structure and Component Design (see Figure 2.2). These levels are thus intended to give full support to designers in a proactive implementation of DFA.

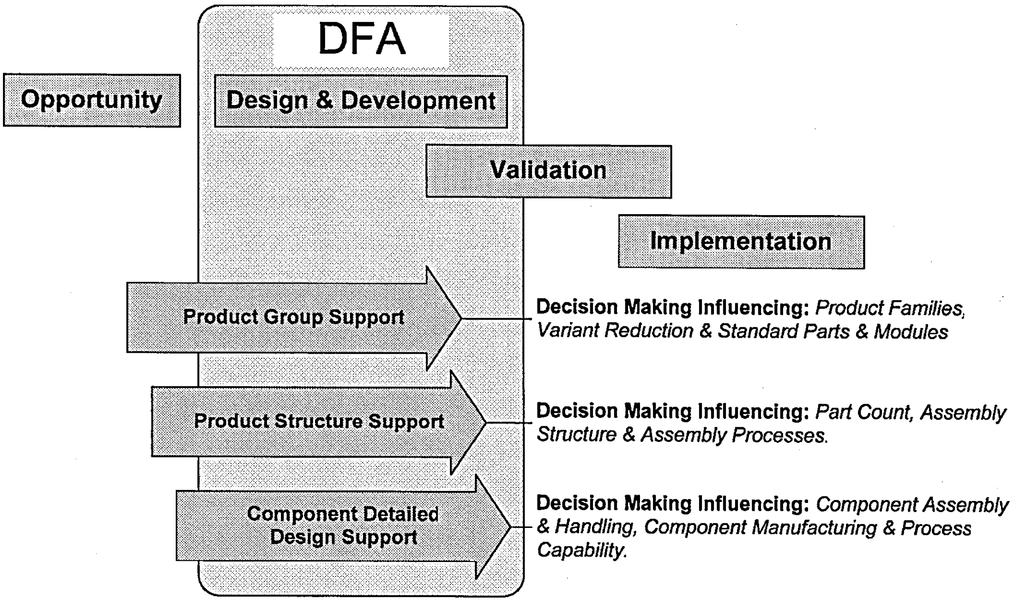

DFA
Opportunity
Design & Development
Validation
Implementation
Product Group Support
Product Structure Support
Component Detailed Design Support
Decision Making Influencing: *Product Families, Variant Reduction & Standard Parts & Modules*
Decision Making Influencing: *Part Count, Assembly Structure & Assembly Processes.*
Decision Making Influencing: *Component Assembly & Handling, Component Manufacturing & Process Capability.*

Figure 2.2 – Implementation of Proactive DFA (adapted from Tate *et al.* [5])

- ***Product Group Support*** – It is conceived with the purpose of identifying product platforms, therefore establishing product families and modules out of common features and similarities. These common features will also be used to encourage component standardisation and rationalisation. Evidently, these standardised components would have to be submitted to DFA analysis beforehand, thus creating an efficient root for the subsequent product families.

- ***Product Structure Support*** – It is proposed to capture the essence of an assembly-oriented design through the evaluation of alternative product structure layouts and assembly sequences. The prediction and indeed the study of suitable assembly sequences have been thoroughly explored in previous works [14] and in the literature in general. The comparison of product structure layouts can be used to evaluate the performance of the product in terms of the functions and the tasks it needs to carry out. This multiplicity of combinations in terms of part count and pairwise interaction is one of the key issues in product complexity analysis.
- ***Component Design Support*** – The final stage of a Proactive DFA implementation is the support of a detailed component design. In order to help designers to generate designs that are suitable for manufacture and assembly, advice is provided for component manipulation, and insertion, but more interestingly, this support bears in mind the shape of the component as being part of a greater structure: the product itself.

2.3.4 Product complexity analysis (an additional area of research)

Despite the interesting advantages offered by the Designers' Sandpit initial proposal, the creation of the suggested Assembly-Oriented Design environment was still heavily dependent on heuristics. A proactive-DFA implementation, as initially presented, was still subjective and error-prone. Tasks as evident and necessary as 'part count optimisation' could not be accomplished on time and when needed (early on in design the process, because there was no way to support the designer on this issue. A tracking system was required.

The author of this thesis considered that it was necessary to implement, within the Sandpit, a system (or software module) that could record and evaluate the product structure, making possible its comparison with DFA guidelines.

It was evident that the Designers' Sandpit lacked a module that could detect, and possibly analyse, the formation of patterns within the structure of whatever product was being designed in the designers' sandpit environment. This detection of structural patterns (i.e. arrays, repeated components, and so on) would be beneficial for an early implementation of DFA guidelines (proactive-DFA for component variation reduction and component standardisation). It would be possible to evaluate the consequences of some of the decisions made by the designer during the creation of the product structure. That is, extracting certain information such as the number of components, number of interactions per component and the types of those interactions, it would be possible to have an estimation of the consequences of those factors in the product's final assembly and manufacture. The extraction of such information required the analysis of the product complexity and the creation of metrics that would help assess it.

2.4 Product Complexity Analysis

Functional modelling, assembly sequence and manufacturability analysis are interlinked in such a way that modifications within the product structure have a direct impact on how the product might work, how it can be put together and ultimately, how it can be manufactured. Product development involves dealing with large number of variables and factors, often factors that contradict each other, such as ease of manufacture vs. ease of assembly. This often means that when certain modifications are introduced, to make the product more viable in one sense, the other (contrary) factor reaches critical levels.

This thesis seeks to explore techniques or means to manage and cope with product complexity. Not surprisingly, Product Structure evaluation becomes extremely relevant. Evaluation of part-to-part interaction and types of interactions, as well as part connectivity, number of connections and part shape are some of the topics examined in this thesis.

The work contained herein addresses some theoretical and (hopefully) practical aspects of complexity evaluation in product design, for a Proactive DFA implementation, such as is that intended in the Designers' Sandpit project. This evaluation ultimately invigorates the collection of tools originally proposed in the project. It adds a powerful tool to monitor and support the work of designers by helping them to create an efficient and DFA-optimised early on in the product development cycle.

Chapter 3

COMPLEXITY AND COMPLEX SYSTEMS: DEFINITIONS

In searching for concise and objective definitions of terms such as 'complexity', 'complex systems' and 'product complexity', one immediately sees that this task seems to be as cumbersome as the concept that they comprise. The published literature is crammed with so many ambiguous and contradictory definitions that academics and practitioners are left baffled. This myriad of definitions and conceptions do not lend themselves to a clear examination of what the terms convey, let alone a suitable form of evaluation. This chapter attempts to shed some light on these multiple definitions, touching various philosophical aspects given to such terms and how they have been considered in several fields. However, special attention will be paid to how they have been interpreted in engineering in order to understand the complexity in product development.

3.1 Definitions of 'complexity'

The term complexity has been unhappily defined as 'Just a Word!', or at least that is the way in which Corning [15] sees it in his article. This author goes as far as simply describing it as 'the least interesting property of complex phenomena.' Whether one agrees with this point of view or not, the author has certainly pointed out that what is in fact interesting are the unique combined properties that arise in every case for any given event, not the commonalities. Equally interesting are his observations about the attributes implied by definitions of complexity:

- It consists of many parts,
- There are many relationships/interactions amongst the parts,

- Parts producing combined effects – often unpredicted, novel, unexpected and surprising.

In the same paper, complexity is defined as a property of the observer, not of the observed system. This situation will be revisited later as the different schools of thought (*See* section 3.5) about complexity are introduced. However, the fact that there is more than one possible definition of complexity draws attention to how pointless it would be to attempt to reduce its metrics to an all-purpose algorithm. As presented in the rest of this chapter, definitions of complexity address all sorts of shades and considerations, but most of them explore the issues of systems complexity or complex systems and there is little about single products, which is the focus of this thesis.

The definitions of Complexity *per se* are as diverse as the world that it involves. A quick browse of the Internet yields thousands of pages related to complexity, ranging from the very strange, up to the most engaging explanations.

These characterisations can be classified according to references dealing with the concept and measurements of complexity. Amongst the sciences that deal with the theory of complexity we can take into account: *Chaos Theory* (very fashionable in the last few years), *Fuzzy logic, Networks, Philosophy, Psychology, Statistics* and so on. The list is enormous and some definitions are somewhat intriguing. These and other references to measures of complexity have been carefully compiled in a bibliography by Edmonds [16]. It follows that there are some considerations of particular relevance to this thesis in sciences such as:

- **Arithmetic** – (a) The number of arithmetic operations needed in a computation – (b) the level in the arithmetic hierarchy.
- **Entropy** – Complexity is defined as the rate at which predictability disappears.
- **Games** – Measures of strategic complexity for games, based on the idea that strategies requiring more detailed information are more complex.
- **Information** – Effective complexity defined as the algorithmic information complexity of an ensemble of patterns.
- **Software** – The "complexity gap" as the region between a top-down specification or planning process and the highest level of software tool available.

In a further paper [17], Edmonds noted that any general estimations of complexity are necessarily abstract and complexity can only be attributed to models of specific processes in any given language.

Herbert Simon [18], considered the father of Artificial Intelligence or 'Complex Information Processing' (as he thought would have been a better name for it) also expressed the idea of complexity being a context-dependent notion. In a series of observations about complex systems and the architecture of complexity, he highlights some common characteristics:

- Complexity of a system depends critically upon how it is described.
- Most complex systems contain a lot of redundancy.
- Simplicity of description can be achieved by finding the right representation.
- Hierarchies of complex systems can often be described in economical terms; that is to say, redundant components can be grouped together and considered as integrated units.

These characteristics of complexity deal with the system as a whole and, as he states, such a system is made up of a large number of parts that have many interactions.

It is not surprising then to see that many complex systems are based on a description that classifies them as *hierarchies*. A complex system defined as a *hierarchy* has the advantage of having a traceable configuration. If a complex system is based on a simple structure with redundant and non-redundant parts continuously added to it, then a finished system is nothing but a chain of built-in features. The lower level is considered the basis and, to build up the system, it is necessary to add redundant, but necessary, extra levels. This idea fits in perfectly with an assembly-oriented design environment where, in most cases, one can find pairwise interaction amongst parts, despite their performing different functions. It also provides the basis for an algorithm to tackle the evaluation of shape complexity of single parts based on a hierarchical arrangement, as discussed in more detail later, in chapter 8.

3.2 Complexity terminology and Complex Systems

Complex system analysis or the "complex systems theory" is said to be concerned with understanding systems in terms of evolution, learning and adaptation, as opposed to the study of complexity. McCarthy *et al.* [19] made this point clear in their work explaining of how complex systems theory has developed within system sciences. In a very efficient manner, McCarthy *et al.* present the progress achieved throughout the ages of complex systems theory. Introducing early rationales about self-organising systems, the authors take the reader through this evolution, starting with the early Cartesian view, through general systems science, cybernetics to finally explain some of the more recent theories in complex systems movement. In doing so, they draw attention to useful and logical observations about complexity, when referring to the work of various scientists such as the physicist Larry Smart, they point out that "... *there is no one right way to define and measure complexity. It is a multi-dimensional, multi-disciplinary concept...*" They also identified some of the most commonly noted hallmarks of complex systems. Characteristics such as *Emergence, Self-organisation, Adaptability* and *Chaos related behaviour* have been shared by many authors, amongst them Jeffrey *et al.* [20], pointing out the links between the fields of biology, physics, mathematics and social sciences in general, with processes and phenomena considered as part of 'complex systems'.

In order to understand the characteristics, descriptions and adjectives given to complex systems, this section presents a compilation of the most commonly used terms.

- **Simple** – A term wrongly understood as the opposite of complex. Simplicity has been recognized by Agudelo-Murguía and Alcalá-Rivero [21] as a complement to complexity. In the sense of complex systems, a system to be regarded as simple has to be easily understood, presenting a fully predictable behaviour.
- **Co-evolution** – A process through which different species (or organisations) adapt and change through interaction with each other and with the environment.
- **Self-organisation** – The spontaneous emergence of non-equilibrium organised structure due to collective interactions. It often emerges from learning, adaptation and innovation.

- **Adaptation** – A term referring to the change process in complex systems. It often results as a response to achieving a certain goal or objective after sensing the feedback given by mechanisms within the system.
- **Emergence** – The appearance of higher-level properties and behaviours of a system that are properties of the whole and not possessed by the individual parts that conform it. This property results directly from the system's evolution and occurs independently, allowing the identification of new opportunities.
- **Redundancy** – The ability of a system to suffer degradation without altering its state.
- **Chaos** – Chaotic systems, although deterministic, have a difficult and often unpredictable behaviour, because future states are sensitive to the current state of the system.

Complex systems are then seen as composed of different types of phenomena carrying out a variety of functions. They are more than complicated systems, for they undergo qualitative and sometimes unpredictable changes. This view is also shared by Hatch and Tsoukas [22]. Some of the conclusions that can be drawn from their report about complexity in organisations extend the previous definitions and can be summarised by five distinctive characteristics of complex systems.

- **Non-linearity** – Complex systems exhibit dynamic behaviour with constant changes that allow them to be adaptable (**dynamism**). As a result, there is no direct proportionality between cause and effect.
- **Fractal-*ity*** – Irregular forms are scale dependent, which means that no single measure will give a true answer, introducing dependence on the measuring device.
- **Recursive symmetry** – Complex systems tend to repeat a basic structure at several levels, such as nested sub-classifications within higher level structures. This is, somehow, an allusion to the hierarchical classification of complex systems.
- **Sensitivity to initial conditions** – Complex systems are unpredictable, volatile and often interpreted as capricious. (Chaotic behaviour).

- **Emergent behaviour** – They are often replete with feedback loops. System behaviour is the emergent outcome of multiple chains of interaction.

As mentioned before, these characteristics can be found in several 'organic' systems and have been studied in parallel with man-made systems, where engineers (scientists and managers as well) have come across several similarities that might help them assess the performance and risk of many systems. In an essay, Ottino [23] describes organisations and systems as working at an optimum state of high-risk, a notion that had been considered by theorists in the area of 'entropic-measures of complexity' [24] and organisations at the 'edge of chaos' [25], a topic that will be further developed below. (*See 3.3 Information Theory and Manufacturing Systems*)

Hatch and Tsoukas [22] based their findings on their understanding about organisations, pointing out a two-tier classification system for complexity. They proposed that theorists ought to move from 'how complex an organisation is' towards 'how an organisation is perceived as complex' following this classification:

- **First-order complexity** – Primarily understood as the complexity of the organisations, that is, an objective perception of how complex organisations really are, and
- **Second-order complexity** – Estimating situations related to the interpretation of the organisations. It is the subjective domain of the thinker reflecting on complexity or the complexity of thinking about organisations.

These divisions also have their roots in what has been considered as the 'schools of thought' about complexity. Nevertheless, this shows the general and tacit consensus reached by several authors when commenting on complexity.

To end this section, some of the vocabulary introduced by Michael Behe[2] in his confrontational book "Darwin's Black Box: The Biochemical Challenge to Evolution" will be considered. He introduces the concept of '*irreducible complexity*' to describe systems (biological or man-made) whereby the removal of any one of the parts causes the system to effectively cease functioning. Another scientist (a

[2] Further reading: Michael J. Behe. (1998) "Darwin's Black Box: Biochemical Challenge to Evolution." Simon & Schuster Inc, New York; ISBN: 0-6848-3493-6.

mathematician to be more precise), William Dembski[3] further enhanced the definition of irreducible complexity by considering that "*such an irreducible complex system is one composed of several well-matched, interacting parts that contribute to the basic function*", hence introducing the idea of 'irreducible core' which is in itself defined as: "*The parts of a complex system which are indispensable to the basic functioning of the system.*" Some of the defenders of the whole idea of irreducible complexity, such as Smart [26], have suggested applications of it in its support. A discussion about such applications, unfortunately, falls out of the scope of the present work, so they will not be described here.

In the context of this thesis, it can only be said that, the concept defies the findings of evolutionists as it introduces the idea of 'intelligent design' as the only way to achieve systems of 'irreducible complexity' in biological systems, leaving no room for the creation of systems formed by numerous, successive and slight modifications. The book has certainly caused controversy and generated passionate discussions that can be followed all over the Internet. However, it has also defined terms that can be useful in engineering (man-made systems), and these are ones that fit in well with the theory presented here, partly helping to define the bottom of the scale in a complexity measure scheme.

3.3 Information Theory and Manufacturing Systems

Most of the literature surveyed shows that the need to deal with complexity has found its roots in management theory. In recent years, there has been an increasing number of publications about how systems should be designed and how complexity should be tackled and measured. A general review of this increasing interest in complex system theory will be given.

[3] Further reading: William A. Dembski. (2001) "*No Free Lunch: why specified complexity cannot be purchased without intelligence.*" Rowman & Littlefield Publishers, Inc., ISBN: 0-7425-1297-5

3.3.1 Tools and theories used in complex manufacturing systems

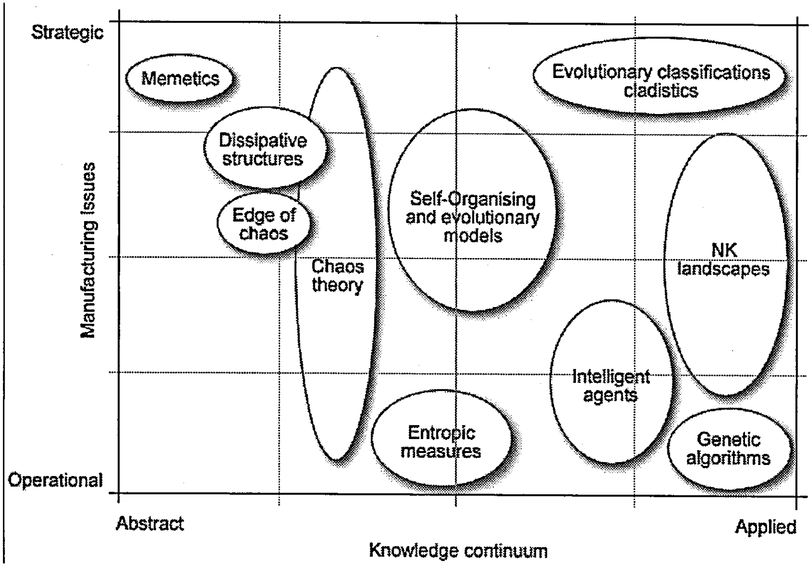


Figure 3.1 – Complex systems framework and manufacturing issues (adapted from McCarthy *et al [19]*)

In an attempt to show how relevant complex systems theory is to manufacturing organisations, McCarthy *et al.* [19] presented a framework that helps to understand how some of the known and emerging bodies of work deal with complex systems. This framework, portrayed in two dimensions as in Figure 3.1, is bound (on one vertical axis) by manufacturing issues, ranging from operational to strategic, and (on the horizontal axis) by the type of knowledge involved, that is from abstract to applied knowledge. Some of these theories and tools are briefly described as follows:

- **Memetics** – The study of 'memes', a made up term from the attempt to characterise the crossing of 'memory' and 'genes', coined by Richard Dawkins[4] in his book "The selfish gene". Dawkins introduced the concept of self-reproducing ideas (memes), which use humans exclusively for their propagation. Memes are to manufacturing organisations what genes are to biological organisms. As briefly explained by McCarthy *et al.* [19], memes are "*like the organisational gene or blueprint, which contains a manufacturing organisation's history ... Thus, one of the main applications of memetics to manufacturing is the*

[4] Further reading: Richard Dawkins, 1989. "*The Selfish Gene.*" Oxford; Oxford University Press. ISBN 0-192-17773-7

ability to understand the 'knowledge management' processes that exist within the organisation .."

- **Chaos theory** – Largely acclaimed as the way to understand the world and see it in a different way. Chaos theory relates to several kinds of irregularity, where it is difficult to predict any future state of the system, given its dependency on the initial conditions.
- **Edge of chaos** – A phase of transition between chaos and order, as the name suggests, it derived from chaos theory. This theory suggests that engineering systems and manufacturing organisations behave optimally at a high-risk state between order and chaos [23], whereby entering this high-risk state organisations can have an optimal performance and high responsiveness to their environment given their self-organising characteristics. Needless to say, this situation puts engineers in a state of unease due to the risks of operating in such a small tract and the possibility of crossing the threshold to disaster. This grand compromise between structure and surprise was described by Kauffman [27] asserting that complexity lies somewhere between chaos and order.

Most of the authors describing "Edge of chaos" theory conclude that it is at this stage where organisms, organizations and systems in general have the highest degree of responsiveness to changes in their surroundings [28].

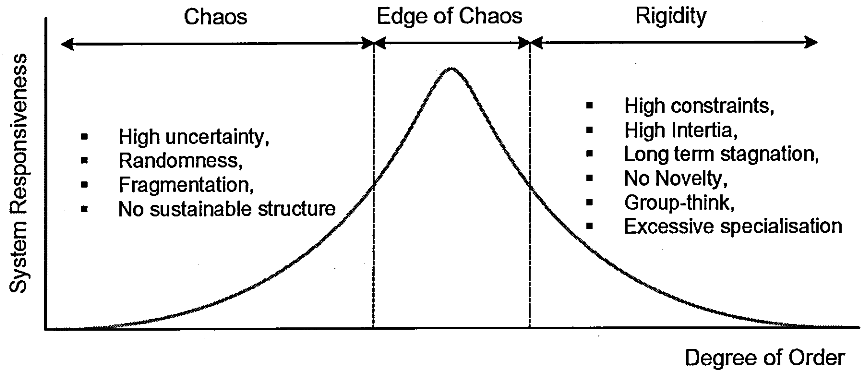


Figure 3.2 – System responsiveness in the edge of chaos (adapted from McCarthy *et al [19].*)

- **Self-organising systems** – Systems which 'evolve' into an organised form in the absence of external pressures. Any systems that take forms that are not imposed from outside can be said to self-organise. Some of the applications commonly associated with Self-organising systems are the so-called 'neural networks', simplified computer models of how the neurons interact within the brain. Since there is not a 'central control', all the 'neurons' that are connected directly or indirectly manage to make sense out of complex patterns of input, hence their self-organising properties. [29]

- **Cladisitics** – This branch of evolutionary biology is a method for analysing the evolutionary relationships between groups to construct their family tree. It has been around for almost fifty years, but has only become widespread in the past two decades. The principle behind it is that organisms should be classified according to their evolutionary relationships, and that the way to discover these relationships is to analyse what are called primitive and derived characters. Rakotobe-Joel *et al.* [30] used this classification technique in strategic management issues, where strategic analysis, strategic choice and strategic implementation where considered key elements.

- **Intelligent agents** –The development of this computational tool has encouraged authors such as Wooldridge [31] to work on and further develop what is also known as 'agent-based technology.' As Wooldridge recognises, there is no universally accepted definition of the term agent, however it has sprung from the idea of organisations and systems in general, where some of the hallmarks of these computer programs are *adaptability*, *autonomous action* and *co-operation* to achieve a common goal. McCarthy *et al.* also realised that the intelligent agent approach clearly places importance on the 'behaviour producing' aspects of a system, rather than 'information structure' aspects of the same.

- **Genetic Algorithms (GA's)** – Genetic algorithms are inspired by Darwin's theory of evolution. The solution to a problem solved by genetic algorithms uses an evolutionary process (it is evolved). An algorithm begins with a set of solutions (equivalent to chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are then selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

- **Dissipative structures** – A phrase coined by Ilya Prigogine[5] as a name for the patterns which self-organise in far-from-equilibrium dissipative systems. Dissipative structures are non-equilibrium thermodynamic systems that generate order spontaneously by exchanging energy with their external environments. According to Prigogine, thermodynamic systems contain subsystems that continuously fluctuate. At times a single fluctuation or a combination of them may become so magnified by possible feedback, that it shatters the pre-existing organisation. At such revolutionary moments or "bifurcation points", it is impossible to determine in advance whether the system will disintegrate into "chaos" or leap to a new, more differentiated, higher level of "order".
- **Fitness landscapes** – The notion of fitness plays a role in self-organised systems, where natural selection is defined by the survival of the fittest. In the case of manufacturing fitness, as addressed by McCarthy [32], Adamides 2002 [33]; Assogna 2002 [34], where at any given time some organisations are more successful than others; they are 'fitter' than others. The fitness of a system changes over time because of the constantly changing environment, which is being remade from moment to moment as an emergent result of the interactions between the systems. This means that a configuration which has a fitness f_1 at time t_1 (relative to the other systems in the environment) is most unlikely to have the same fitness at time t_2. The notion of 'fitness landscape' was originally proposed in the 1930's (Swell Wright) and further developed by Stuart Kauffman [35]. Kauffman suggests that in a dynamic, changing environment a snapshot of each setting would reveal a 'landscape'. If all points (organisations) are equally fit, then the landscape would flat. In the event differences in fitness between the organisations appear, the landscape would reflect a hilly profile, and if one system is much fitter than the others, then the representation will be like peaks rising from a valley.

[5] Further reading: Nicolis, G. and Prigogine, I. (1977) "*Self-organization in non-equilibrium systems: from dissipative structures to order through fluctuations.*" New York: Wiley; ISBN: 0471-02401-5

- **Entropic measures** – This method is based entirely on the mathematical theory of communication. This idea is developed in the following section.

The work done by Claude Shannon "A mathematical theory of communication," published in two parts in 1948 editions of the Bell System Technical Journal [36] and later on released as a book [37], has been long regarded as a classic of the modern digital era of communications. Shannon's work on 'information theory' motivated Frizelle and Woodcock [38] to propose entropic-measures of complexity. They started working on adaptive systems such as manufacturing systems, trying to understand their behaviour in terms of the information such systems generated. The amount of information emitted by these entities - either by the number of components, processing lots or the total of operations required to create a product – has been taken as a parallel to energy dissipation as studied in thermodynamics. Although manufacturing systems are man-made organisations, they behave in similar ways to organic systems. They adapt themselves to meet the requirements of customers, the expectations of the market, government regulations and in general the need for producing better and more efficient products. This constant adaptation or search for equilibrium with their environment yields a by-product: information. In order to understand the level of adaptation, the observer needs to capture and process it accordingly to be able to make decisions such as where resources need to be allocated or energies directed. This constant emanation of information has been called "entropy of the system" and it is thought to measure the 'amount of information required to understand the system.' Frizelle [24] presented his findings in a work entitled: "The management of complexity in manufacturing". His work generated a, now popular, subdivision in complexity analysis, namely: static and dynamic complexity.

This subdivision can be clearly seen in the work presented by Calinescu *et al.* about manufacturing systems [39]. Their study and work are based on the aforementioned *entropic measures of information*, divided, precisely, into static (structural) and dynamic (operational) aspects of complexity. Their paper presents the complexity of a system as directly proportional to the '*amount of information needed to describe the state of the system.*' The structural complexity accounts for the resources and their state; such a measure can be used in scheduling those resources. The dynamic complexity, on the other hand, accounts for the behaviour of the system and is used for monitoring a manufacturing facility. The main attraction of this work is the compilation of theoretical and practical measures of complexity in manufacturing systems, giving particular emphasis to the entropic-based formulae suggested by Frizelle and Deshmukh. This latter author presented his work on static complexity, which will be considered below. Finally, Calinescu *et al.* proposed a methodology for measuring the complexity of manufacturing systems and their supply chains. However, because the research is directed more at management of the manufacturing processes, rather than the details of the processes themselves, it cannot be directly related to the product design, as required by our research for the Designers' Sandpit.

Along the same management-science-oriented lines is the work presented by Huaccho-Huatuco *et al.* [40]. They suggested a model to illustrate the interaction between the production and the scheduling functions. Such a model takes into account the regular monitoring of the system in order to check how closely it follows the schedule and to react accordingly if it has deviated from schedule, finally suggesting some rescheduling strategies for managing manufacturing systems complexity.

In order to understand the subdivisions given to complexity by the previous authors, the next two sections will briefly describe the differences and similarities between static and dynamic complexity.

3.3.2 Static complexity

The framework for this categorisation was presented by Deshmukh *et al.* [41] in their work about the analysis of static complexity in manufacturing systems. Deshmukh *et al.* define static complexity as *"the measure of information needed to describe the system and its components, where by 'components' the authors imply all the elements of manufacturing system required to make the selected set of parts"*. Whereas Frizelle [38], defines static complexity as *"the measure of the intrinsic difficulty for the process of producing the required number and type of products in the required period of time"*.

Deshmukh *et al.* considered an entropy-based formula to describe theoretical observations about manufacturing systems. The manufacturing environment they considered represents a discrete part manufacturing system, with multiple part types being machined or formed in the system simultaneously. Static complexity in manufacturing systems is a function of the structure of the system, the variety of sub-systems, and the strengths of interactions. They proposed a series of conditions to be met by such an entropy-based metric, namely:

- Static complexity should increase with the number of parts, number of machines and the number of operations required to process the part mix.
- Static complexity should increase with increase in sequence flexibility for the parts in the production batch.
- Static complexity should increase as sharing of resources by parts increases.
- If the original part mix is split into two or more groups, then the complexity of processing should remain constant.

The term has also been used in the description of product complexity, but it relates to the instant view of the product configuration, such as a snapshot of its architecture at a particular time.

3.3.3 Dynamic complexity

Also referred to as "Operational Complexity", Dynamic complexity was originally proposed by Frizelle [38] with a view to following the operational behaviour of a system from direct observations of the manufacturing process.

Frizelle defines Operational (dynamic) complexity as *"a measure of the operational behaviour of a system base on direct observations of the process, in particular on how queues behave (in terms of queue length, variability and composition)".*

Dynamic complexity is an entropic measure that suggests that operational complexity is reflected by queues, thus, it would help to pin point the causes of obstacles or bottlenecks in the process. These obstacles are regarded as being generated either by internal sources (control of the facility) or external sources, such as customers and market behaviour. Frizelle defines dynamic complexity as

This measure is aimed at looking after process stability and, as a result, controls the system dynamism. Operational complexity can be controlled to improve schedule observance, although it does not lend itself to planning for. It is precisely the work by Frizelle on production scheduling that Efstathiou *et al.*[42] extended with the creation of two expert systems to guide production managers, planners and schedulers on how to assess their facilities for complexity, and identify the problems which prevent them from achieving and maintaining cost-effective flexibility. The same research group [40] developed a generic model to illustrate the interaction between the production and scheduling functions. This model is mainly targeted at rescheduling if necessary; it does this by regularly monitoring the system checking for its adherence to the schedule and reacting accordingly.

However, dynamic complexity has also been observed in a different light. Adamides [33] examines this measure to suggest the development of emergent manufacturing strategies. Adamides was inspired by previous works on the behaviour of systems exhibiting high dynamic complexity. These studies were focused on feedback loops containing stocks (levels) and flows (rates).

Frizelle at the research group at the Institute for Manufacturing in Cambridge University and the Manufacturing Research Group in Oxford acknowledged that, although the foundations of their work rely on entropic-based approaches to measuring complexity (such as that of static and dynamic complexity), there is not a general global multi-purpose methodology for measuring all aspects of manufacturing complexity – an opinion shared by the author of this thesis. However, they present a classification and definition of complexity based on time, control and processes [43]. This new classification is made up of three categories: *decision-making, structural* and *behavioural complexity*. The reasons for this new sub-classification derives from the rationale that '*static vs. dynamic*' definitions are not complete and that they need to include more sources of complexity. The author of this thesis believes that although their intention is that of showing that there is not a single measure of complexity, they have created three new categories that (seemingly) would account for all the sources of manufacturing complexity; in other words, they exchanged a two-class approach (static vs. dynamic) for a three-category approach. This is something the author of this thesis does not consider a great improvement to measuring complexity, as it simply adds another layer of abstraction to the evaluation of complexity, but it does not elaborate on a suitable methodology for complexity evaluation.

The work carried out by these research groups reveals, nonetheless, that perhaps the best approach to tackling complexity is by devising separate measures of it which, if used in conjunction, will create a set of values that will support the people in charge of making decisions during the design stages.

Incidentally, the expressions *static* and *dynamic complexity* are used later in this thesis for the two sub-categories of product complexity. However, these are used in a different sense. The term "static complexity" refers to the product structure studied at a given time. It is as if a 'snapshot' of the product architecture was taken and its complexity level analysed on the spot. On the other hand, the term "dynamic complexity" denotes an ongoing process, and therefore this is the position taken in this thesis. Dynamic complexity, as seen in this work, attempts to encapsulate the intricacies derived from assembly sequence evaluation, component cluster formation and parallel assembly (whenever possible).

3.4 Modelling and learning in complex systems

The unpredictability and dynamism of adaptive systems have raised the need for a tool that will help scientists and engineers understand and try to predict behaviour. Building models of the system has been the most likely choice.

Modelling is an activity commonly used to represent processes or objects at a much smaller scale. These representations can later be used in calculations supporting projects in full scale or in a more convenient form. However, modelling complex systems has an altogether different approach. Usually their representation is not intended for the recreation of the systems themselves, but for gaining an understanding about how the systems might behave and recognising sources of complexity and hence, critical factors.

In 1967, Arthur Koestler [44] presented some of the earliest works related to the art of modelling complex systems recounting the parable suggested by Simon – who later on published it on his book "the sciences of the artificial" [45]. Koestler, as well as Simon, recognised the importance of the creation of hierarchies in organisations in order for them to avoid "chaotic" behaviours. Koestler, nevertheless, extended his view with the introduction of a new concept: Holons. A holon, as defined by him, is a model-component with a 'Janus-face[6]' characteristic. This entity is considered a basic unit in biological and social systems. The word *Holon*, is a combination of the Greek work *holos* (*whole*) and the suffix *on* (*particle* or *part*, as in neutr*on*, prot*on*, etc.). Koestler observed that in living organisms and in social organisations entirely self supporting, non-interacting entities did not exist. The Janus effect of holons is presented in the hybrid nature of sub-wholes/parts in real-life systems; holons simultaneously are self-contained wholes to their subordinated parts, and dependent parts when seen from the inverse direction. Holons are, therefore, continuously keeping track of their upper levels (whole), yet maintaining their independence (self-contained) when handling contingencies without asking higher authorities for instructions and looking after their inferior holons, all at the same time.

It is said that one of the strengths of this holonic organisation (*holorchy*) is that it enables the construction of very complex systems, efficient in the use or resources and highly resilient to disturbances, yet flexible enough to adapt themselves to changes in the environment in which they exist.

Holons or Holonics have been developed a little further by the scientific community. Manufacturing industries/systems have embraced it, adopting concepts from living organisms to attain benefits such as stability in the face of disturbance, adaptability and flexibility in the face of change, and efficient use of available resources – as highlighted in previous sections.

As the reader might have realised by now, holons can be seen as the foundation for intelligent agents. Intelligent agents, or manufacturing agents as seen previously in the work by McCarthy *et al.* [19], have been considered as being independent in their operations and behaviour. These agents are characterized by internal states (operating, idle), input reception (process material), output generation (send information) and some of the decisions they make (machining, scheduling,

[6] In Roman mythology, Janus was the god of gates and doorways, depicted with two faces looking in opposite directions.

ordering, etc.). Intelligent agents, as machine learning techniques for managing complexity, represent the core of the work presented by Monostori [46]. Monostori recounts a survey of some of the artificial intelligence techniques currently used by scientists and engineers. Techniques such as pattern recognition, expert systems, artificial neural networks and fuzzy systems have now been combined in hybrid systems as multi-strategic machine learning approaches to managing complexity. Agent-based systems, according to Monostori, are one of the most promising tools for managing complexity, changes and disturbances in production systems.

Some of the computational tools mentioned above have lately supported the modelling of adaptive systems in order to understand their behaviour. Manufacturing systems lend themselves perfectly to these sorts of studies. The aim of these computational tools is the recreation of systems and, in doing that, learning about their inherent complexity. Modelling complex adaptive systems in such a way produces the knowledge necessary to create and recreate planning scenarios. These scenarios will be the source of information that ultimately supports the people in charge of making decisions (designers, engineers, managers, directors, etc.). Van der Heijden [47], in his book about scenario planning, wrote: "*Human beings and organisations do not act in response to reality but to an internally constructed version of reality.*" He was referring to the models created to understand adaptive systems. Van der Heijden pinpointed, what he considers, the three most distinctive approaches to act in response to that pseudo-reality, describing them as the three "schools of thought" in strategy:

- **Rationalists** – Their premise is based on the principle that there is a unique answer and, therefore, the main purpose is to get to that answer whatever it takes. It is derived from the idea of a well-defined future, where situations can be mapped one-to-one; therefore it is susceptible to variations that may arise during the process. Its linear approach makes it suitable for short-termed projects.

- **Evolutionists** – Their strategy is more accommodating than the previous one, for it takes into account the development of the system. It is based on a retrospective view of that system. It succeeds in taking into account the unpredictability of the system by acknowledging emergent behaviours and situations at a much larger scale, than the rationalistic approach, but because it is mainly reactive and founded on the avoidance of constraints to the systems, it lacks the power to confront new situations much quicker.

- **Processualists** – Their view serves as a compromise between the two previous perspectives. It considers that most systems are too complex to analyse in their entirety, as the evolutionist approach does, but the processualists consider small steps for learning to control the systems, hence considering the rationalistic approach for shorter projects. Rationalists and Evolutionists do not care too much about the process in the middle, as long as there are or are not answers, Processualists, on the other hand, are keenly interested on the process itself.

According to Van der Heijden, these approaches ought to be integrated in an organisational environment and in continuous interaction. Based on the work by several authors, he suggested the application of these approaches within what is called the "**learning loop**". This situation is applicable to individuals as well as organisations and companies. In short, this integration "loop plus strategies" is as follows: the 'rationalistic' approach takes 'observations and reflections', creating and 'forming abstract concepts and theories', the 'processualistic' view considers the 'testing implications of these theories in new situations', to finally produce 'concrete experiences' that will be understood by the 'evolutionistic' method generating new 'observations and reflections' and thus completing the loop.

Inspired by these ideas, Lyons *et al.* [48] worked on models of complex systems for strategic decision making, adopting the processualists' approach as the foundation of their work. They acknowledged that the development of models must not be considered as the end itself, but as part of a wider decision-making process. Consequently, they highlighted a list of constraints (levels) to the modelling problem, namely: (Level 5) Setting boundaries, (Level 4) Choice of frames, (Level 3) Problem structuring, (Level 2) Interpretation of the structure and (Level 1) Assignation of referents. Lyons *et al.* declared that the higher the level the more abstract it is, hence when moving from higher to lower levels, there is a narrowing of focus. The models created, using previously described computational tools (agent-based methodologies), will allow users to test their hypothesis in terms of investigation of alternative control strategies and understanding the implications of specific actions.

Despite having presented some of the existing tools (computational tools) for helping with the construction of and the type of strategies used in complex system models, knowledge acquisition and learning are still two essential topics. Addressing these matters presents at least two unsettling questions, explicitly: (1) *how can relevant information be produced and presented?* And (2) *how can this information or knowledge be used more conveniently?* Authors such as Allen [49] and Jones *et al.* [50] have decided to tackle these questions, and in doing so, they highlighted some critical points regarding information and complex systems performance, as well as some of the reasons as to why people can make decisions in spite of products and organisations not being as obvious as they seem.

Allen quite rightly asserts that knowledge is perceived within a paradoxical state, when he writes: "… *knowledge is something that 'favours' action … but if knowledge is the fruit of learning, learning is the fruit of experience and ,experience is the fruit of actions, therefore action precedes knowledge …* " Indirectly, he is referring to the 'learning loop' as suggested by Van der Heijden [47]. Allen underlines the importance of homogeneity and stability in an environment if knowledge from one organisation were transferred to another. This can only be possible if the situations in which they work are comparable.

3.5 Schools of thought

It is worth mentioning how complexity is to be considered in this thesis. The aim is to measure complexity, thus it should be described or defined first. Such definitions tend to fall into one of two major schools of thought, namely:

a. *Structural-based definition (property of the object):* Complexity considered as a property of the object (system, product, etc.), which depends on the number of parts, part interaction, possibilities in their interactions and other characteristics.

b. *Information-based definition (property of the observer):* Complexity considered as a property of the subject (observer), that is, it depends on the description of the system given by the observer, specifically the language used and the number of different descriptions given [51].

Although it is difficult to ascertain which of these is more appropriate to this application, the first option – complexity as an intrinsic property of the system – has been followed. It is probably the most suitable and is, as might be expected, less subjective to measure, thus making the control of the complexity of a product also less subjective.

The following chapter focuses on complexity in product development and, particularly, in the methodologies for product design optimisation or so-called DFX methods.

Chapter 4

COMPLEXITIES IN PRODUCT DEVELOPMENT

The written definitions and terminology related to complexity offered an insight of the fields where it has been studied. Nevertheless, they have been mainly oriented to the study of large systems, either in the area of management or manufacturing systems. Even though manufacturing systems are directly concerned with building up products, the study of such systems does not contemplate the development of products as the main target.

Some authors have proposed models of product technology that describe products as complex systems [52], arguing that products as well as systems are made out of elements with specific characteristics to function collectively according to how these elements are put together. However, the author of this thesis suggests that one crucial difference between products and systems is that products are finite[7]. Products are, by themselves, complex entities. They are often perceived as the source of an entire system of manufacturing, transport and economic evolution. Products are end results, whereas systems, such as manufacturing systems, are open-ended entities exposed to enhancements, updates and expansions. These observations are presented in Chapter 5 and are partly based on the findings presented in this and the subsequent sections.

Some considerations given to complexity in the development of products have generated 'conceptual definitions', which are more often encountered in design, as well as quantitative methodologies, which are frequently related to product manufacturing. For the purpose of this thesis and without loss of generality, the

[7] The clause: "*A product is a system in itself*" can be seen under the notion of components interacting geometrically and functionally within a self-contained entity: A product.

studies reviewed have been classified into: conceptual (design) and mathematical models of complexity. An additional section on quantification of shape complexity has been presented to emphasise the original intention of this work: manufacturability analysis.

Product development has often been depicted as a cycle that starts with the identification of an opportunity. This leads to thinking about how the product will solve a particular set of requirements (concept design), producing an initial outline of the product working as the core around which engineering data is continuously fed in (detailed design), tested and validated. Finally, once all evaluations have produced satisfactory results, the product plans are released for manufacturing. This extremely simplified description of the cycle contains a great deal of decision-making situations that add to the product design complexity. It has also led to the study and evaluation of the human psyche and the influence that it has in the complexity of the product, especially in terms of its decision-making [53] and design-parameter coupling [54] capabilities.

Design complexity is closely linked to an increase in design effort as shown by Bashir and Thomson [55]. There has been an increase in the number of studies and endeavours to provide measures of complexity and cost effectiveness for systems design and products, as presented by Guenov [56], who described two of the most generalised views of design: *architectural* and *axiomatic design* [57] (both of these will be revisited). Such a division, nonetheless, is suitable for studies of complexity as it draws attention to two independent, yet interwoven, categories: ***structural*** and ***functional complexity***. Guenov's work is reinforced with entropic measures of complexity, but he limits the scope of design complexity by saying "*... one can decide which [design solution] is more complex on the basis of measurable physical quantities such as number of parts, connections, supply chain size, and so forth ...*" Although this is not wrong, it certainly needs more explanation, which is the intention of this and the subsequent sections.

Since the main focus of this thesis is to produce complexity measures for the proactive implementation of DFA/M methodologies, it necessitates taking a closer look at the ***structural complexity*** of the design. Needless to say it does imply overlooking functional analyses. Structure and function cannot be separated, but it is convenient to focus attention on one or other aspect. This type of (functional) analysis will be mentioned when necessary, but the central point will be structural or architectural. For this reason, the following sections will present the results of scanning for such information in the literature.

This search evolved into a two-way examination: *component*-based and *assembly*-based analysis. Component-oriented analyses are mainly directed towards their construction, either as solid models (geometry and topology) or as how they are manufactured, which leads to evaluation of geometric shapes.

For reasons that will become clearer as the discussion evolves, shape analysis was considered to be the primary goal of this research, but it then expanded to a more holistic approach involving all attributes and characteristics of assemblies.

4.1 Component-based analysis

Complexity in design is generally considered in relation to component geometry where it has been studied for its influence in many areas. This research can be classified according to the following categorisations: geometry, topology and assembly.

4.1.1 Geometry

In applications such as computer graphics and finite element mesh generation, polygon meshes are defined in terms of the geometry (coordinate values of vertices of the meshes making up the model) and connectivity of the object (the relationship amongst the vertices that define the polygonal faces of the mesh) [58]. Several algorithms for triangulation of the mesh have been used and, in some of these, shape complexity is a factor for determining the resolution [59]. Informally, this notion of shape-complexity measures how entangled a polygon is.

Other studies have examined the nature of data exchange through the Internet and between different modelling platforms [60, 61]. Compression of the model topology involves reducing redundant references to vertices and edges that are shared by many entities. The compression of geometric data requires the precision of coordinate values, normal vectors, etc. to be reduced and this is often achieved using techniques from signal processing. This is required in order to reduce file sizes. Model complexity in this context is used to establish the optimum compression algorithm for the particular model to ensure a minimum loss of information.

Another overlapping consideration of complexity in design is that of "CAD Complexity". In a study by Chase and Murty [62] not only is the geometric complexity of the CAD model considered, but also the complexity of the CAD file system of organisation and the operational complexity of the CAD software in terms of its functionality. The notion of a hierarchical CAD file system corresponds well with the traditional model in design of a product structure tree.

4.1.2 Topology

In 1981, Kyprianou [63] pioneered feature recognition techniques for the purposes of classifying shape for automated part coding, but since that time, feature recognition has been applied to many aspects of design, and different measures of complexity have been established depending upon the precise application.

Part Coding is based on the manufacturing philosophy of Group Technology Part Codes, in which similar parts are identified and grouped together for the purposes of generating manufacturing plans and scheduling production. The process is automated using CAD data as input [64] and enables re-use of existing manufacturing process plans or provides a starting point for generation of a new plan. Some systems provide a 'fuzzy' search facility based on an index of 'similarity' between the required set of features and those recorded for each part in its catalogue. The similarity index is based on a comparison of part codes and considers both features in common and those that are absent [65].

The 3Dsearch.net project at Heriot-Watt University uses algorithms to assess the geometric similarity of 3D models and applies it to the Internet sourcing of engineering components [66-68]. Within this and similar work, considerations such as the number of faces in a model, number of sides of a polygon, curvedness, symmetry, number of turns, degree of compactness, angular variability and crinkliness have all been used to compare aspects of shape complexity [69].

Yet another interpretation of complexity is aimed at the feature recognition research community themselves, to compare the effectiveness and reliability of different feature recognition algorithms. One such algorithm is presented by Little *et al* [70] who attempt to generate a complexity code for an object in terms of the number of face types: their relative orientations and interactions (local complexity) and their distribution across the object (global complexity). Psarra studied the influence of local properties to analyse and describe shape complexity in architecture [71]. Using 2D architectural drawings, Psarra explored the emergence of patterns in several shapes, which seemed to represent contour stabilities and differentiations.

4.1.3 Shape complexity in support of DFX methodologies

Measuring shape complexity is not a new concept; it is not even enjoying a new boom now that computer power has increased. It has been thoroughly explored for 2D objects [59] and for considerations of shape similarity for image retrieval systems [72], but it has not been fully explored in 3D models and even less in the DFA environment.

Some complexity factors have been suggested for specific processes, such as the one proposed by Tomov [73]. He presents a new shape complexity factor for forged parts based on FEM analysis and considerations of the amount of volume transformed between two arbitrary states. Despite its quantifiable nature, the factor is not suitable for DFA and, therefore, it is not useful for the intended scope of the Designers' Sandpit environment.

On the other hand, Kim *et al* [74] obtained a fully quantifiable factor and, even better, it is DFA oriented! Their factor is based on the feedability of parts, which is in itself based on old stochastic parts feeding algorithms reported in the same paper. This approach consists of a feeding mechanism (a device with a gripper that can orient parts under software control) and the planning algorithm. The algorithm recognises the grasping plans based on geometric properties, namely their polygonal contour. The variables considered are the initial and final orientations. Since the initial part orientation is uncertain, thus the first actions consist of selecting random orientations in order to grasp the part. This process is repeated until the part is correctly oriented. The mathematical model is based on probabilities and transfer functions that take the first action as a probabilistic distribution of all potential orientations. The feedability of the part is based on the number of steps necessary to orient the part and reach a plan with stable grasping orientations. In spite of its attempt to provide a shape metric of the part, this technique has two major drawbacks. Firstly, it is based on probabilistic measures and leaves no room for a geometric reasoning approach. Secondly, it is dependent on the orientation of the part to find its symmetry and try to match that of the gripper. This condition of 'orientation required' has been re-evaluated and dismissed by Tate [75] in studies of symmetry detection using geometric reasoning.

Thus considerable further research is required to make complexity a practically useful concept within the Designers' Sandpit environment. (See *4.4* Complexity in the Designers' Sandpit Environment)

4.2 Assembly-related analysis

Although the focus of 'complexity' research in design has been in relation to component geometry, some attempts have been made to quantify the complexity of an assembly. Goldwasser [76] defines 'virtual assembly sequencing' as the separation of specific geometric assumptions from the assembly sequencing, turning it into a graph-theoretic problem focused on the combinatorial aspects of feasible assembly sequences.

Similarly, Wilson and Latombe [77] define product complexity as the lower-bound complexity of all feasible assembly sequences. The complexity of each assembly sequence is evaluated in terms of the insertion trajectories, the number of hands required and the gripping configuration during component assembly operations. The number of assembly operations is also considered.

Due to its relevance, assembly complexity will be one of the major aspects of analysis and study in product design and development; therefore an entire section of this thesis will be dedicated to it (Chapter 6).

4.3 Specific complexity metrics

The methods described in the preceding section do not address specific measures of complexity. However, in some disciplines more precise metrics have been developed. Software engineering has demonstrated that the evaluation of the complexity of algorithms is paramount to establish the performance of a particular piece of software. The most important criterion to analyse is generally the running time [78], where the type of algorithm used and the inputs to it play a major role. Nevertheless there are several other factors that affect the running time of a program, such as compilers and type of computer, but they are not considered during theoretical estimations of algorithm performance. Although software is not considered a tangible, physical and discrete product or at least not in the sense taken by other areas of engineering, its analysis presents some similarities to its material counterparts. McCabe's early studies of software complexity measures produced a theory called *cyclomatic complexity* [79], which was later on used to derive a set of quantifiable design metrics [80] These design metrics, namely: *module design complexity, design complexity* and *integration complexity* make use of graph representations to express their approach. McCabe's approach focuses on 'control flow' complexity and does not take into account the data as a contributor to complexity. A survey on and a comparison of several software metrics was reported by Riguzzi [81]. These and other software metrics have inspired the scientific community to produce similar metrics in product engineering, but taking account of differences between software analysis and product development.

4.3.1 Design complexity metrics

In order to achieve a quantitative evaluation of the design process, Maimon and Braha [82] presented a mathematical analysis of the complexity of design, where the design activity was perceived as the process of mapping specifications (goal space) and the solution space. Their work soon developed into a method to analyse a model of the design process, more specifically the synthesis of the design process. Therefore, they called their method the "Basic Synthesis Problem" or BSP [83], whereby they acknowledged the difficulties in the tractability of the design constraints proper of an evolutionary process carried on in cyclic fashion.

Despite being apparently more conceptual than concrete, in terms of numerical values, Suh [84] defines complexity in the previously mentioned field of *axiomatic design* [57, 85] as a "*measure of uncertainty in achieving the specified FRs.*" Suh relates complexity to the information content (I) which he defined as a "*logarithmic function of the probability of achieving the Functional Requirements (FR)*" of a design.

Suh later on subdivides complexity into *time-dependent* and *time-independent complexity*, and as if that were not enough, there is a further subdivision for time-independent complexity into *real* and *imaginary complexity*, both completely orthogonal and forming a vector sum of absolute complexity.

The real complexity is regarded as the uncertainty of fulfilling the functional requirements of a design. Once the problem has been defined, the design parameters form the set of factors that will be used to minimise the real complexity. As soon as the real complexity has been reduced, the design is expected to meet as many functional requirements as possible, given the appropriate definition of the design parameters. Imaginary complexity, on the other hand, is held to be the lack of understanding of the system designed, giving the impression of a complex system when in fact it is not.

The numerical relation between the functional requirements (FR) and the design parameters (DP) is given by the probability of finding the right combination of any DPs to satisfy the entire set of FRs. The complexity (real complexity) is given as a function of the information content for satisfying a number of FRs. El-Haik and Yang [86] developed this concept further, discussing the mathematical representations of Suh's axioms and, based on entropic measures, calculating complexity of an engineering design. Nevertheless, in both cases, their approach is focused on an overall complexity estimation of the design where vulnerabilities (related to the size of design problem) and variations in the functional requirements play a greater role.

Finally, Suh's time-dependent complexity defined as dependent on the uncertainty of future events that cannot be predicted *a priori*. This type of complexity, which is not clearly related to product design, will not be extended in this thesis. Sadly, this leaves the time-independent complexity as the first approximation to define complexity metrics in designing systems which, as mentioned above, does not satisfy initial requirements of our research due to its overall methodology: that is, the calculation of complexity is estimated for the whole of the design problems, failing to highlight specific sources of complexity.

Nevertheless, complexity has been studied in other types of systems. In the area of manufacturing systems for instance, as mentioned above, Calinescu *et al* [39] have surveyed the field of complexity for a more tangible definition of it and also proposed some formulae for its assessment. (*See Chapter 3, Entropic measures*).

Another view of systems complexity is that given by Tang [7] in adaptive design. In this paper, the author stresses the lack of theory, foundations and software to support complexity metrics. He proposes a qualitative differentiation between '*architected complexity*' and '*complicatedness*'; the first one being a beneficial property of systems provided it reduces such complicatedness. He devises a formula in which complexity is a function of the number of elements, their interactions and the *bandwidth*[8] amongst the elements of the system. This function increases monotonically, provided the system size, interaction of elements and bandwidth increase as well. Complicatedness, on the other hand, is the measure of the capacity of the system to handle its own complexity. As a result, complicatedness is a function of complexity with an asymptotic behaviour until it reaches a point of saturation; thereafter the system enters a stage of unmanageable complicatedness. This growth pattern commonly occurs in fields such as biology, engineering and economics.

Another overall complexity metric was proposed by Braha and Maimon [87]. Based on their previous work [83], they presented a new measure by following the principles set by Nam Suh on his 'axiomatic design' theory, whereby Braha and Maimon attempt to measure the '*information content of a design*' and, therefore, the design's complexity. Braha and Maimon were motivated by the general idea of 'design simplicity', where a good design is said to have been obtained if it satisfies all its functional requirements, with the minimum number of components. This is a well-understood principle followed by many, including the author of this thesis.

This time, however, Braha and Maimon founded their work on the measures of 'software complexity' initiated by Maurice Halstead [88] as well as the seminal work of Claude Shannon [36] on information theory (which was mentioned in the last chapter). They, Braha and Maimon, worked on the two main hierarchical

[8] Alas, in this context the definition of bandwidth is rather vague.

decompositions of the design process: structural and functional. They defined the latter functional complexity by defining the information content in a functional way, and therefore "*as a function of its probability of successfully achieving the functional requirements or as the probability of success*". Its mathematical formulation is surprisingly similar to the one proposed by Nam Suh [84] and therefore it will not be commented upon. Finally, Braha and Maimon defined the *structural complexity* of the design as a function of its representation. They followed almost exactly the token-based approach proposed by Halstead, whose method proposed the measure of a programme as a collection of tokens, classified as operands or operators. Halstead's method was a measure of the '*volume of a programme V*', to which Braha and Maimon called the '*structural information content H*'. All the similarities and metrics are presented in Table 4.1.

Table 4.1 – Halstead's vs. Braha-Maimon's metrics

Halstead's metrics	Braha-Maimon's metrics
$\mu 1$ – number of unique operators	ρ – number of unique or distinct operators appearing in the design form
$\mu 2$ – number of unique operands	N – number of unique or distinct basic operands appearing in the design form
N1 – total occurrences of operators	N1 – total number of occurrences of the operators in the design form
N2 – total occurrences of operands	N2 – total number of occurrences of the operands in the design form
Length of *P* : (P = computer programme) $N = N1 + N2$	Length of the design form $L = N1 + N2$
Estimated Length: $\acute{N} = \mu 1 \log 2\, \mu 1 + \mu 2 \log 2\, \mu 2$	
Vocabulary of *P* $\mu = \mu 1 + \mu 2$	Alphabet of the design $\eta = \rho + N$
Volume of *P* $V = N \times \log 2\, \mu$	Structural information content $H = L \times \log_2 \eta = (N1+N2) \log 2\, (\rho + N)$
Potential Volume $V^* = (\mu 2 + 2) \log 2\, (\mu 2 + 2)$	Minimal information content $H^* = L^* \times \log_2 \eta^* = (2+N^*) \log 2(2+N^*)$
Program Level of *P* $L = V^*/V$	Design abstraction level $A = H^* / H$
Estimated Program Level $\acute{L} = 1/D = 2/\mu 1 \times \mu 2/N2$	
Difficulty of *P* $D = 1/L$	
Effort Required to generate *P* $E = V/\acute{L} = \mu 1\, N2N \log 2\, \mu / 2\mu 2$	Designing effort $E = 1/A \ \times H$
Time Required to generate *P* $T = E / S$ S = number of mental discriminations per second S = 18 for programming	Time complexity measure $T = (1/S \ \times A) \times H = E/S = H^2 / (H \times S)$ S = rate at which the brain makes elementary mental discriminations (in seconds) S varies between 5 and 20 (often used S=18)

From Table 4.1 it can be seen that Braha and Maimon made a one-to-one use of the metrics proposed, in 1977, by Halstead for his studies on software complexity. In 1980, Smith [89] showed that the estimated length proposed by Halstead could give an idea of the size about the programme, but he could not determine anything else. However, a report presented in 1996 by Riguzzi [81] dismissed the rest of the metrics Halstead proposed, as lacking experimental demonstration and being found in a non-scientific manner.

Halstead's work produced an early attempt to measure the complexity of software. These metrics have been criticized and used by other computer science practitioners; some suggested that counting operators and operands has not been generally agreed upon by researchers. Whatever the reasons for approving or dismissing Halstead's measures, it can certainly be said that Braha-Maimon's approach can be criticised because of the way they made use of such metrics. Their approach is a direct one-to-one use, failing to make any distinction between the difficulties emerging from software programming and those of mechanical artefacts design. Finally, another clear point of blunt similarity can be perceived in the use of the suggested rate at which "the brain makes elementary mental discriminations (S)," Braha and Maimon used the value of S = 18 as suggested by Halstead for software metrics.

Conclusively, one can see that despite having a logical approach to measuring complexity, namely, information content, the metrics proposed for structural complexity are quite dubious. Furthermore, they represent a measure taken as a whole, failing to pinpoint specific sources of complexity.

4.3.2 Complexity metrics for DFX methods

It is well-known that DFX methods provide useful guidelines to obtain efficient designs, but these benefits often produce contradicting outcomes. A typical example is the trade-off between DFA and DFM, the first suggests that the ideal design should have as few components as possible, but still be able to fulfil all functional requirements. This, more often than not, implies the integration and combination of parts that, although easy to assemble, are hard to manufacture. On the other hand, DFM suggests part/component simplicity and to achieve this, components are likely to be like building blocks, easy to manufacture, but too many to offer a good assemblability rating. These DFX tradeoffs are one of the major concerns of this thesis.

Striking a balance between DFA and DFM is crucial to be able to get the benefits offered by these two methodologies. This task often relies on the knowledge of experienced designers and even they consider this to be a difficult enough job, let alone the introduction of other DFX tools. Authors such as Kalsi *et al.* [90] have reported on the difficulties of decision trade-offs in complex systems design.

There have been some attempts to incorporate metrics in DFX methods. Metrics that attempt to support users and designers with methods capable of highlighting critical values that may hinder the benefits offered by specific tools. These indicators have been developed for areas such as: *Product quality and reliability, Product recyclability and disassembly* and *Product modularity and variety*. It has also been found that there have been some early attempts to introduce suitable indicators of complexity factors in DFA. These findings, and the indicators proposed, are described as follows:

- ***Design for Reliability*** – Reliability and quality measures are two indistinctive terms often used in product design to address (the avoidance of) defects in manufactured units. Beiter *et al.* [91], as well as Shibata *et al.* [92], appointed the work of Hinckley as part of an "assembly quality method". Hinckley's studies [93] were presented in 1993, where, by use of time studies, he hypothesised about the relation between product complexity and the number of expected defects and its utility as a method of assessing a product's assemblability. Hinckley based his analysis on the Westinghouse DFA worksheet [12], which suggested TAF or Total Assembly Factor (also called Total Assembly Time - TAT). TAT represented the 'theoretical time required for product assemble.' Hinckley then took the 'Total Number of Operations' and multiplied by the Ideal Assembly Time (t) to assemble a single part and subtracted the product from TAT to yield the value of the "Complexity Factor", as seen on Equation 4.1.

$$Cf = TAT - (t \times TOP)$$

Equation 4.1 – Hinckley's Complexity Factor

The ideal assembly time is based on a part being picked-up, inserted downward into the assembly and being symmetric end-to-end. This complexity factor attempts to measure the excess time required to complete the assembly, where a complexity factor of zero would imply an ideal assembly sequence for the entire product.

Hinckley suggested that separate complexity factors can be generated for multiple products assembled at the same facility. Figure 4.1 shows the correlation between the number of defects per unit and the complexity factor. Hinckley also suggested that this can be used as 'benchmark quality control' independent of the product complexity over a variety of conditions [94].

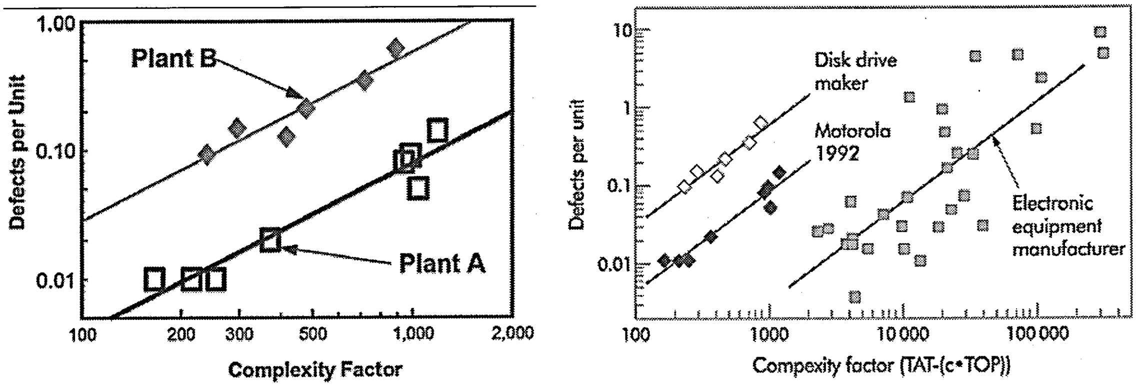


Figure 4.1 – Hinckley's Assembly defect vs. Complexity Factor

It is clear that Hinckley's factor heavily depends on statistical data, which as expected require that, at least, one version of the intended product has already been manufactured, for defect rates to be recorded.

- ***Design for Recyclability, Reusability and Disassembly*** – These methods relate to the last stages of the product life cycle, where a robust design can help cut product retirement costs and scrap through the appropriate selection of design methods. For that matter, Lee *et al.* [95] proposed a *recyclability map* which highlights the difficulties of sorting and material recovery efficiency for major subassemblies. This set of complexity measures are described as follows:
  - o *Disassembly complexity* – This measure is based on the cost of the disassembly process. Lee *et al.* use what they called a "reverse fishbone

diagram (RFD)." (Figure 4.2). The RFD is simply a hierarchical representation of the structure of the product, where every level is defined by performing, at least, one disjointing operation. As in many hierarchical representations, Lee et al. estimated that size[9] and shape of the reverse fishbone indicated the complexity and cost associated with the de-manufacturing process. The RFD assumes that at every level a single part (represented by a circle) will be obtained and disposed of in a particular 'sort bin'. The disposal procedure can be either 'scrap' or 'recycle'. Lee *et al.* argue that the total disassembly cost heavily depends on the number of sort bins. Despite this description, a quantifiable metric was not provided.

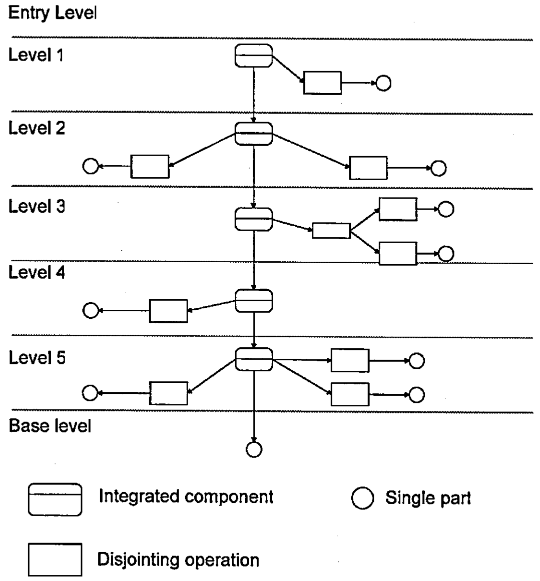


Figure 4.2 – Reverse Fishbone Diagram (RFD) (After Lee et al., 1997)

- *Sort complexity* – This metric captures information about the difficulty and costs of the disassembly process as influence by 'level of recycling technology employed' and 'product reuse/re-manufacture'. The '*number of sort bins*' used during the disassembly procedure would indicate the level of complexity. Sort bins are defined as any distinct end destination for a product, module, subassembly or component. Lee et al. estimate that

[9] It is not clear whether the term 'size' relates to the number of levels alone and no particular importance is given to the breadth of the hierarchy.

the higher the number of sort bins the deeper the levels of disassembly, higher material count and low commonality[10].

- o *Material complexity* – This metric in particular refers to the number of materials used in a component, subassembly or single part. Considerations are given to *the number of material classes* (i.e. plastics, ferrous and non-ferrous metals, paper and wood, hazardous materials, other), *materials compatibility*, *special handling* (reverting to the differences presented by typical DFA methods) and *the relative valuation of materials*.

The metrics proposed are similar to those offered by DFM methodologies (i.e. material selection). The hierarchical structure proposed or RFD falls back on considering straight vertical decomposition of a structural hierarchy to account for the number of levels. The authors, Lee *et al.*, recognised the limitations of their methodology, but at least it can be considered as foundation for future work.

- ***Design for Modularity*** – This deserves a section on its own, due to its direct association with structural and functional complexity. (See Section 4.5.3 '*Product modularisation and variants: Design for Modularity*')
- ***Design for Assembly*** – This method main guideline is simplification through part-count reduction. Boothroyd and Dewhurst introduced the "complexity factor" [96] of a design as the measure of the complexity of a design. It allegedly provides a numerical indication of the relative complexity of an assembly. They acknowledged that a single value is misleading unless used to support comparison with alternative solutions.

Boothroyd-Dewhurst complexity measure accounts for three different variables, i.e. *number of part types, number of parts* and *number of local relations* and the complexity factor can be calculated as follows:

[10] Component commonality is often used in Design for Variety methods, see Section 4.5.4 '*Product variety, change and flexible manufacture: Design for Variety*' for a further development on this argument.

$$Cf = \sqrt[3]{(Nt \times Np \times Ni)}$$

Equation 4.2 – Boothroyd-Dewhurst's complexity factor

Where:

Nt = Number of part types – Each new part can be taken as a new part type.

Np = Number of parts – The total number of components in the assembly

Ni = Number of local relations – Number of binary pair-wise interactions. Expressed in this manner, binary interactions can be counted twice (once from every component's point of view).

This measure needs to be applied consistently to allow comparisons between design solutions. But it still serves as a rough estimator of the assembly difficulty. However, it presents some pitfalls. Balasz [97] explored this metric and encountered a lack of robustness, for it can be tricked into producing the same complexity measure for two arrangements clearly perceived as non-similar. Balasz presented a counter-example using a simplified version of Boothroyd-Dewhurst's factor, where the type of components was not considered. This 'simplified' metric resulted in the *square root of the number of parts times the count of local relations* as shown in Equation 4.3.

$$Cf' = \sqrt{(Np \times Ni)}$$

Equation 4.3 – Balasz's simplified version of Boothroyd-Dewhurst complexity factor

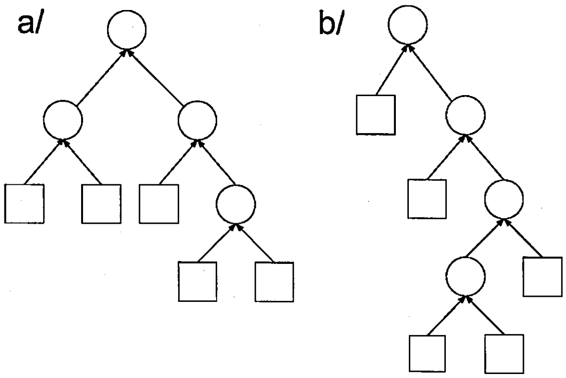


Figure 4.3 – Two relation hierarchies with the same Boothroyd-Dewhurst's complexity factor value (After Balasz)

Balasz [97] illustrated, with an example similar to the one depicted in Figure 4.3 (where *circles represent relations and squares represent components*), that Boothroyd-Dewhurst's metric was built under the assumption that there are no higher level relations in the structure (i.e. relations amongst relations), therefore, it did not distinguish between designs with the same number of components and same number of relations, but using higher level relations. Balasz presented his example as follows: Using the hierarchies depicted in Figure 4.3 and Equation 4.3, one can see that for hierarchy type (a) $Cf' = \sqrt{((Np = 4) \times (Ni = 4))} = 4$ and for hierarchy type (b)$Cf' = \sqrt{((Np = 4) \times (Ni = 4))} = 4$.

Certainly, one can find counter-examples to contradict most of the metrics proposed. In order to avoid misleading results, complexity metrics must be used in conjunction to form a set of metrics that can account for the several factors that introduce complexity into the design.

4.4 Complexity in the Designers' Sandpit Environment

The design process supported by present CAD tools tends to be component-oriented rather than dealing with a product as a whole. This leads to redesign, rework and a lengthy product introduction process. Design evaluation tools, such as Design for Assembly (DFA), have demonstrated some success in reducing these problems, but these can only be applied once a complete product description is available and hence the potential benefits of early analysis are not realised. A more proactive approach to DFA [98] requires support for assembly planning during the design stage and this is one aspect of the research that the Designers' Sandpit seeks to address.

The Designers' Sandpit environment is based on the idea that a large proportion of all product costs are determined at the design stage and much of these costs are incurred during assembly. In fact, there is evidence to suggest that, in industry, products are still designed with at least 50% [99-101] excess of parts and assembly content and undergo more complex assembly procedures than is necessary. Thus, the designer needs to consider assembly whilst developing product design in order to mitigate subsequent problems on the shop floor. It is now accepted that product assembly considered at an early stage of the design, promotes study of the design as a whole, which has been proven to improve overall costs, quality and time to market. However, many current commercially available CAD packages still tend to concentrate upon 'component-oriented' design, where individual parts are modelled and then assembled to create the final product. Hence, a need has been identified for development of an "assembly-oriented" CAD environment. As mentioned above, DFA claims proven success in reducing part count, improving product quality and minimising assembly problems. Therefore DFA, integrated within a CAD environment, has been established as a significant step for consideration of assembly issues.

One aspect of the DFA methodology [11] is an evaluation of the manufacturing processes required for each component of the product, based upon a subjective shape complexity analysis, reminiscent of part coding. However, as parts are integrated or eliminated to optimise assembly with respect to part count and assembly operations, inevitably more complex components are created and more complex insertion processes are required. The overall assessment requires that the time and cost of production is minimised, but this obviously requires a compromise between manufacturing overheads, assembly processes and part count.

This is where the role of geometric reasoning is much appreciated. In order to evaluate a product with respect to all these aspects of design it is important to consider complexity in many different ways. Ideally, the calculation of shape complexity should be automated using data extracted from the CAD model, to reduce subjectivity and enable a direct mapping between a component and its ideal method of manufacture. Furthermore, an appropriate classification scheme would also enable reuse of existing parts and the reduction of variance, both within the product and across a range of products, by making component comparisons possible. A measure of complexity for the assembly and product configuration should also be considered. The ultimate objective is to fulfil the product function by creating the 'simplest' design.

4.4.1 Role of Complexity Analyses

Striking a balance between sources of complexity after following DFA/DFM methods needs to be placed in context. Such context could be explained within a hierarchical structure that would help to evaluate and monitor changes in the product architecture.

Nevertheless, it has proved hard to put in context the notion of complexity, for there is a generally held view which may be expressed as: "no-one has ever succeeded in giving a definition of complexity which is meaningful enough to enable one to measure exactly how complex a system is". If the goal were that of producing a single value that would convey the idea of complexity, then there would be, indeed, no way to know exactly how complex that system is. Products cannot and should not be reduced to one single complexity metric. Products are not only the end result or solution to a given opportunity; but also the source of an entire system of manufacturing, transport and economic evolution.

Therefore, if instead of a suitable definition, that accounts for all that is known to produce complex systems, a description of it were achieved, then complexity could be measured, monitored and mapped into what complex products will generate. A classification of the concept, such as the one depicted in Figure 4.1, provides the framework necessary to monitor sources of complexity. Factors that can be traced and recorded, thus keeping historical data of the design process, will help in understanding the process itself. Recording historical 'know-how' is not at all new, it has been thoroughly reported by Hatch and Tsoukas [22]. It has been thought of as a means of considering the different bifurcations encountered in (design) processes, whereby the interpretation of point C implies knowledge of the history of the product in the system, which had to go through points A and B. Furthermore, when this kind of data has been recorded and plotted as the development process evolves; it is easier to spot the change that originated an increase in the overall complexity rating.

In order to find the optimum balance for a product design, between manufacturing capabilities and assembly operations, the notion of complexity will be considered at two levels: Assembly and Component. A general layout of these two levels of complexity is shown in Figure 4.4.

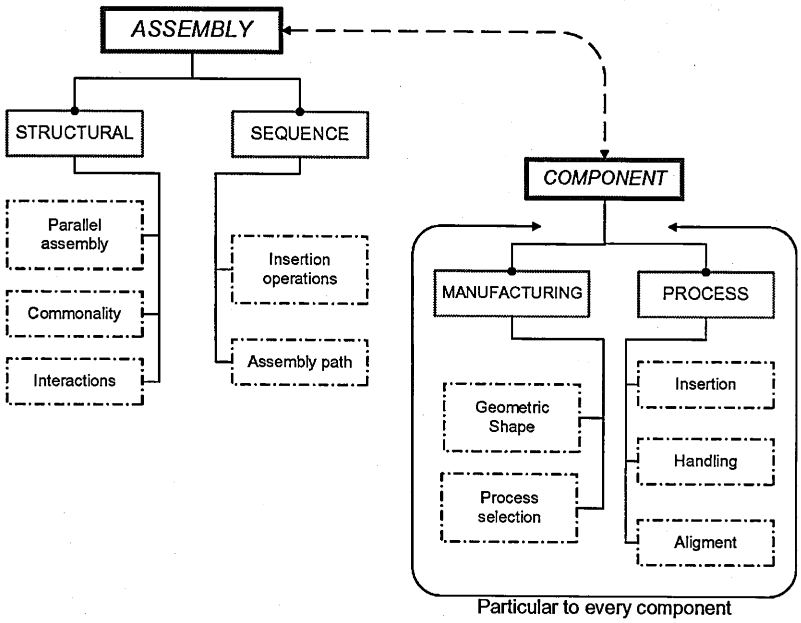


Figure 4.4 – Different complexity types considered at a particular level.

4.4.2 Assembly Complexity

Assembly complexity encompasses those aspects of a design that affect the efficiency of the assembly sequence, component interaction and the efficiency of the whole product design. It has been properly considered as the activity that actually creates the product.

Time spent in putting one product together is often considered as the major descriptor of its complexity. Human factors are known for their influence in assembly performance due to manual assembly. Nevertheless, psychologists [102] have also studied the influence of external causes on the assembly operation, highlighting tasks variables such as (a) selection of components, (b) symmetrical planes, (c) fastening points and fastenings, which are dependent on the number of components, component groups and novel assemblies (that is, new assemblies required more time to learn and absorb the assembly procedure).

The above mentioned variables have already been studied for years, with most authors agreeing on the fact that when products are assembled, activities such as design, manufacturing, manipulation and logistics converge [103]. Assembly is also the activity that dictates most of the complexity of the product itself.

Complexity of the product is, accordingly, more than the sum of the complexity of the components, this being a holistic approach to product complexity identification. Moreover, most of the aspects or variables known for being the source of assembly complexity or simply to be considered during assembly can be classified into two large groups:

Structural Complexity, C_{st} – The configuration of a product in terms of its product structure is not addressed by current DFA analyses, other than to eliminate non-functional parts wherever possible. However, the structural breakdown can have a great impact on the ease of assembly, but more particularly on the critical assembly path. Subassemblies within a product structure enable parallel processing on the shop floor and provide more flexibility for production scheduling. Proper consideration of such issues also enables use of modular design techniques in some industries. However, a hierarchy that is too 'deep' generates increased part tracking, storage and inspection requirements. Part handling also becomes significantly more difficult since the interfaces between subassemblies require careful consideration to minimise problems when mating groups of parts.

Sequence Complexity, C_s – The number of operations required to assemble a product is directly influenced by decisions made at the design stage, but also by the assembly sequence chosen. The number of insertion operations is directly proportional to the number of components, but a badly defined assembly sequence may incorporate many unnecessary operations, such as turnovers, and necessitate additional tooling.

Sequence analysis and assembly planning have been two of the most extensively studied topics in the design of assemblies. Assembly sequence has been considered as part of integrated CAD systems [104-106], based on graphs and operational networks [107-109], pattern matching [110] and its support with artificial intelligence [111, 112].

Conveniently, Barnes *et al.* [113] have presented a group of indices to assist designers in evaluating assembly sequence quality, this work will be further discussed and considered in the chapter 6, dedicated to assembly complexity.

4.4.3 Component Complexity

Component complexity encompasses those aspects of the design that relate directly to each component and are not directly affected by the assembly sequence. These are:

Manufacturing Complexity, C_m – The type, number and difficulty of manufacturing operations is directly related to the geometry of a component. Some attempts have been made to automate the calculation of shape complexity, as previously discussed. However, the existing DFA methodology requires the designer to subjectively select a shape classification according to a global shape type (rotational, prismatic or thin-wall) and a complexity ranking based on the number and type of additional features. This is used to identify components with particularly inefficient or costly manufacture. As the number of parts is reduced to improve assembly, this can result in yet more complex components, requiring new materials and/or manufacturing methods. Care must be taken to ensure that the benefits of reduced part count are not outweighed by the cost of producing the new components.

Process or Manipulation Complexity, C_p – This notion of complexity is required to quantify the difficulty associated with alignment, insertion and handling operations on individual parts or subassemblies. Current DFA techniques provide a scoring system to evaluate these aspects of assembly and thus highlight components that present particular problems. Alleviation of these problems through redesign is often also beneficial in terms of manufacturing complexity.

4.5 Further issues for product complexity assessment

The main target of this research has been the study of spatial complexity in the context of a proactive implementation of the DFA methodology. The central purpose of the DFA analysis is the improvement of assemblability of existing parts or their manufacturability. Nevertheless, engineers must bear in mind the function of the parts within the product. Authors such as Tuttle [114] have suggested a Design for Function (DFF) methodology which might help the designer to establish the design criteria for a product, translating customer functions into product functions. It somehow would share common grounds with *Axiomatic Design [57, 85]* and *QFD* [115, 116] methodologies. But DFF, unlike the other two, was proposed as a means of reducing the number of parts, that is, it might be employed at later stages of the conceptual design phase, when products have a somewhat defined architecture. It, nevertheless, did not assess the complexity of the product's functionality.

Spatial complexity somehow tackles the problem of *where* a part connects to, and *how* it does it in space, which is why it is more easily associated with hierarchies. Nevertheless, there are 'functional' hierarchies which emphasise processes in time, that is, with what parts in a product *do*. Thinking of product functionality as a hierarchy is somewhat abstract, but it makes sense if it is seen as a chain of triggering effects. For instance, the designer will specify a required functionality (functionality at level i), however, some design solutions may have mandatory elements (e.g. a hydraulic system will always require a 'bleed' function/capability, an electronic solution will require actuators, etc.), which could be added as sub-functions (functionality level i+1), and in this way a functional hierarchy is created.

However, designers are still on their own when assessing functionality, for they are responsible for estimating the percentage contribution of each part or substructure to the functionality of the product. This is the same as to say, that they still face the task of mapping a set of functions or performance targets (functional hierarchy) to a product structure (spatial/architectural hierarchy) as seen on Figure 4.5. The difficulty of this task increases when a single component may perform many functions, or conversely a single function may be performed by many components. Several mapping representations, such as 'bond graphs' [117, 118], have been suggested to display the links between function and form.

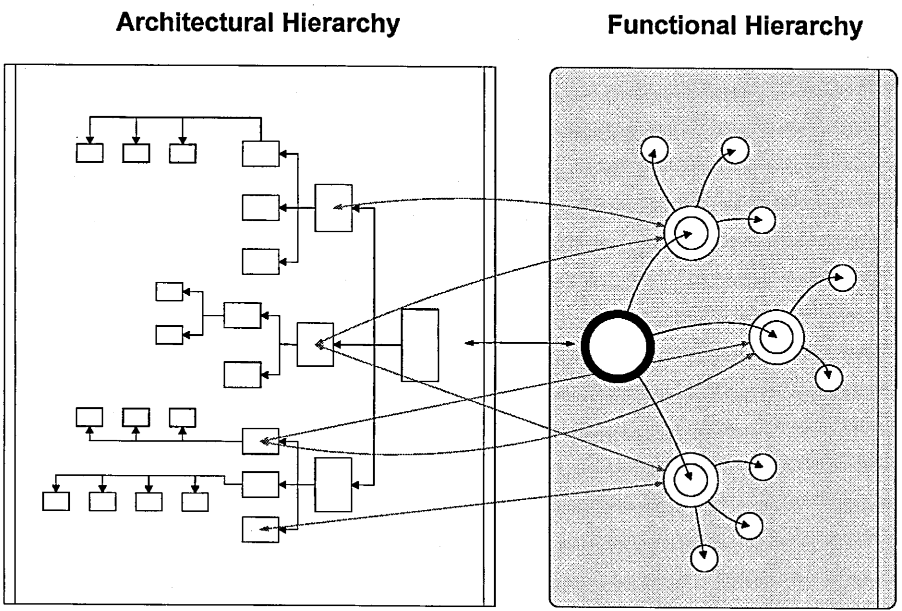


Figure 4.5 – Architectural and Functional Hierarchies (Function-to-part mapping)

In the following sections, the subject of functionality and its assessment will be briefly presented, so one can have an idea as to how it has been tackled by the academic community.

4.5.1 Functional complexity: Design for Functionality

Design optimisations as well as the selection of appropriate design solutions are based on an optimal representation of the functional requirements of the product/design. This representation constitutes one of the basic requirements of a system intended to assist the designer in generating and evaluating alternative concepts. However, where product 'functionality' assessment is concerned, there are several notions that make the definition of 'function' an elusive idea. Some identify 'functionality' as expressed in terms of constraints and kinematics. Others propose that function and functionality should be asserted based on whether a part contributes to the requirements of a design (design intent or purpose).

In order to evaluate product functionality and, therefore, the complexity associated with it, the creation of a new vocabulary or 'design grammar' was proposed by Andersson [119]. He suggested a particular hierarchical configuration that would integrate design concepts, requirements, physical principles and geometric elements, amongst others. It is, consequently, a result of the concept design stage, where function-to-form mapping is expected to happen. Andersson chose to describe the 'characteristics' of both functions and parts (or 'physical principles, as he called them) to get around the problem of a direct connection between them. As he clearly put it, a language for design is not a new idea, but such work was presented as a means of representing syntactic and interpretative knowledge, in other words, it followed the notion of creating a part to fulfil required design behaviours.

In terms of functional complexity, Andersson relates it to design effort: " ... *the more complex the wanted design is, the greater the effort needed to ensure that the selected solution principles can be composed into an assembly that satisfies the whole function model* ..." Similar conclusions can be read from the work done by Bashir and Thomson [55, 120, 121]. The authors relate the amount of time and effort invested in design projects and, hence, the design of products to the complex functionality involved. Based, once again, on a hierarchical configuration, Bashir and Thomson proposed a functional decomposition, breaking down every function that the product to be designed must perform into sub-functions, and if possible, these sub-functions into yet more sub-functions, until a level of basic functions is reached. Finally the sum of the number of decomposition levels (or tree structure depth) times the number of functions at a specific level is presented as the functional complexity index of the product:

$$PC = \sum_i (f_i \times i)$$

[*Product Complexity = sum (number of functions at the ith-level, times the ith-level)*].

Before one can jump to any derogatory conclusion, they acknowledged that the metric involved a degree of subjectivity, which required standard protocols for such functional decomposition. However, the protocols recommended are a tautology in their own right. They suggested that the lowest levels of decomposition must meet conclusions such as: *(a) It was mapped to a component which was designed by a subcontractor, (b) It was mapped to an existing component and (c) It was considered simple*. One can see that these are somewhat weak arguments; conclusion (a) is perfectly possible, but it is a get out assumption. Conclusions (b) and (c) just turn around the subjectivity of the functional decomposition, but it remains subjective. Despite such an ineffective argumentation, this approach is still valid, the hierarchical methodology is consistent and the index, although simplistic, serves as a first intuitive suggestion of the functional complexity. Eventually, this index can be added to the collection of metrics to be used in conjunction to highlight the overall complexity of the product.

Finally, Sivaloganathan *et al.* [122, 123] introduced the "Design Function Deployment. (DFD)" As its name implies, it is related to Concurrent Engineering and the Quality Function Deployment (QFD). DFD is acclaimed by its authors as a prescriptive system that combines the benefits of QFD, Concurrent Engineering, Design models and methods, and Computer Aided-Engineering. DFD is in fact a prescriptive system, for it regulates, through a set of charts, the application of the methodologies it combines, offering the means to rate designs and store them in charts for future comparison. But it failed to address aspects different to those tackled by the methods it combines.

4.5.2 Functionality interlink with structural decomposition

On the other hand, it is not surprising to know that if functionality is defined as a process dependent on time, then kinematical analyses actually take over the functional aspects of a product. Those inseparable spatio-temporal characteristics of product design were followed by Mantripragada and Whitney [124], who introduced a concept called the "Datum Flow Chain" (DFC) to capture kinematical constraints and part positioning. DFC serves as the constraint plan for getting the parts to the right places in <u>space</u> so as to deliver characteristic requirements of the product. These specific requirements where given the name "Key Characteristics" (KC) by Lee and Thornton [125]. These KC's are intended to be issues such as critical performance, safety, or regulations. Lee and Thornton stated that KC's must be used during concept design to capture customer requirements, during detail design to deliver requirements via tolerances and process planning and during program management to track and assure the achievement of those requirements. Accordingly, KC's should be confined to things that are not only important but are at some risk of not being achieved. Thornton later on defined a mathematical framework of the Key Characteristic process [126]. Despite their specification of kinematical characteristics, this approach is more towards structural complexity, than functional.

The 'Datum Flow Chain' is represented as a directed acyclic graph of an assembly with nodes representing the parts and the links (referred to as 'mates' or 'contacts'), drawn as 'arcs' to represent the type connections between them. These connections are presented to deliver a particular set of 'Key characteristics', namely tolerances. A similar approach was attempted in this thesis, where the type of connections is considered as a 'weight' to specify the importance of the connection/interface.

This thesis also develops a matrix that shows the relationships between components (e.g. their interactions and interfaces). A similar idea was presented by Eppinger and Whitney [127] called the 'Design Structure Matrix' (DSM), which represents the relationships amongst elements. However, the latter is oriented towards the structuring of large projects, which are held to be complex on account of the large amount of components and interactions within. As expected, the DSM uses a matrix in order to capture both the sequence of and the technical relationships among the many design tasks to be performed. Those interactions, or "technical structure", are analysed to produce an alternative configuration, in some cases suggesting a sequence of those tasks. The DSM has been used as a means to restructure projects and help organisations to coordinate the interactions among groups of people within technical projects [128]. Although, it does not particularly address component interactions, it has been referenced in this thesis to stress the difference between the methodologies used in the DSM method and those addressed in this thesis – where a matrix-based representation method was used to study component-to-component interaction and interface types. This, as expected, will be developed in later chapters

4.5.3 Product modularisation and variants: Design for Modularity

It has been seen that some of the complexities in product development are attributable to the product's in-built architectural composition and to the tasks it must accomplish. However, there are several additional conditions, imposed by customers and the product market, that have created further types of complexities.

Market trends have shown that: (a) Customers are becoming more demanding in terms of product variety and short development times. This tendency has created a whole new concept of mass production turning into *mass customisation*[11], driven mainly by the increase in global completion. (b) Customers want products that can be stripped of components they are not interested in and at the same time, products that can be more accommodating of customer choices.

This newly added variable of extra options within the same product opens up the chances of new types of complexities cropping up during product development. The very process of planning the partitioning of the product into different options or variants is a complex task on its own right. Estimating what component(s) will serve as the product base or platform and thus defining the subsequent product families inherently crosses the line between spatial/architectural aspects and functionality characteristics of the product itself.

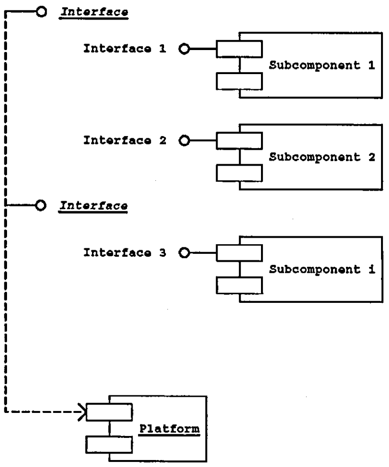


Figure 4.6 – Modularisation

The question is: *why would a designer want to undergo such a traumatic and convoluted practice*? Well, the answer seems to lie in the life-cycle of the product [129-132]. Designers can utilise product similarities, reusing components and introducing standardisation, which helps them lower the risks of new product introduction by using well-tested and known solutions. Therefore, the benefits gained from the independent manufacturing of units (work in parallel, distributable tasks), separate and independent testing, as well as the possibility of having interchangeable components seem to outweigh the design efforts and complexities involved with embracing product modularity [133, 134].

The life-cycle of the product not only defines the way in which a company/manufacturing firm addresses the tasks involved in product development, but also the architecture of the product. This is the main conclusion read from the

[11] A concept introduced in 1993 by B. Joseph Pine in his book: "*Mass customization: the new frontier in business competition.*" Boston. Harvard Business School Press; ISBN: 0-875-84372-7.

seminal work on Product architecture[12] by Ulrich [3]. Ulrich addresses the issue of modularity by clearly stressing the point of mapping functional elements to physical components and in the process highlighting the trade-offs between modular and integral architectures. Modular architectures are defined as a one-to-one (function to component) mapping, as opposed to the integral architecture where there are one-to-many or many-to-one mappings. Consequently, he defined the vocabulary that would help trace some of the complexities in product design: *Interface definition*. Definition of interfaces between any two modules and amongst components in general, involved *geometric connections* (a concept strongly used in this thesis, see Chapter 5), which required contact (e.g. pinion-gear) or non-contact interactions (e.g. infrared communications). Following these principles Erixon suggested a method called 'Modular Function Deployment (MFD)' [135] based on a set of different criteria used to group functions (also called 'module drives') according to strategic aspects such as financial constraints, product diversity, technological evolution, stylistic aspects and planned changes, amongst others, which he stresses, are some of the factors that lead a Design for Modularity [136]. Erixon acknowledged the difference between interfaces dealing with 'energy exchange' (e.g. signals, forces, electricity) and 'geometric aspects'. Sadly, his acknowledgement did not go any further, leaving the problem of interface complexity to be measured by the assemblability metrics proposed by the Boothroyd-Dewhust DFA methodology [101]. In other words, he reverted the problem of complexity estimation to a subjective assessment of it.

Differences between the types of interfaces can make the change between modular and integrative products as reported by Sosa *et al.* Modular products are said to have well-defined and identifiable interfaces, whereas integrative products (or systems as they put it) present interfaces that are shared across the product [137]. They make use of the 'Design Structure Matrix' (DSM) [127] to emphasize such difference and the interactions amongst different designing teams within an organisation. In a similar fashion, Van Wie *et al.* [138], on the other hand, worked on the principles presented by Ulrich and found that the number of interfaces and their complexity directly affected the final product costs. However, regardless of

[12] The term "product architecture" often refers to the physical characteristic of a product, whereas Ulrich uses it for both physical (*what components are*) and functional (*what components do*) aspects of the product.

these findings, Van Wie *et al.* did not provide any methods for estimating such complexities, other than the use of the 'Design Structure Matrix' (DSM) [127]. Likewise, Gu *et al.* [139] presented a methodology to identify a set of factors related to life-cycle objectives, relating these objectives to design components through interaction analysis. Such analysis included the inspection of functional interactions pointing out the factors such as exchange of material, energy, signal and force and spatial or geometric interactions that included factors such as attachment and relative positioning. Finally, factors related to maintenance such as frequency of failure of the components, repair similarity and complexity and the down time for repair. Despite acknowledging the incidence of the complexity of component interactions – in this case, for maintenance– Gu *et al.* failed to explained how such complexity must be assessed.

Alternatively, Holtta and Otto [140] presented a method based on heuristics. They asked a group of experienced practising design engineers to rank a series of modules in terms of (a) time spent in redesigning such module in both a larger and smaller scale, and (b) time spent in incorporating such redesign into the original product. It is interesting to analyse their approach under the concepts learnt at the beginning of this section and the definitions and methodologies studied in the last chapter. The methodology presented by Holtta and Otto required the identification of the customer needs and, consequently, the tasks (functions) to be performed by the product, thus creating a *functional hierarchy* – although they called them *function chains*. These chains are meant to be taken as 'customer-critical flows', which produce 'function structure flow', such flow is monitored across the different component interactions and its *intensity* is recorded. This intensity is measured in terms of energy (electrical, mechanical and/or pneumatic) as well as information and material flows. Finally, they used this flow intensity as an indicator of interface difficulty, that is, impediment to redesign it, hence its complexity. They concluded that those interfaces with lower flow intensity are candidates for a module boundary, whereas the highly intensive interactions should be kept within a module.

The criterion proposed by Holtta and Otto should be considered as another useful metric to monitor the complexity of a design and should be included in the collection of complexity metrics to be used in conjunction with others presented in the rest of this thesis.

4.5.4 Product variety, change and flexible manufacture: Design for Variety

As seen in the previous section, product variety has generated a new trend in engineering design, that is, *product modularity*. However, product variety, as understood in this section, is dependent on customer requirements as opposed to the ease of design and manufacturing presented by modularisation.

Ulrich [3] also addresses the point of product change and relates product change to the transformations and variations that the product might suffer during its particular life. In product change one should consider aspects such as *product upgrade, add-ons* (third party or same company), *adaptation* (e.g. engine conversions petrol to propane fuel supply), *wear, consumption and flexibility in use*.

MacDuffie [141] identified five types of complexity in manufacturing performance due to product variety in the automobile industry. These complexity types fall into two major categories as shown in Table 4.2.

Table 4.2 – MacDuffie's Product complexity measures in product variety

Fundamental Variety	Peripheral Variety
Model Mix Complexity	Option content
Part Complexity	Option variability (2 distinct values)

Fundamental variety, as its name implies, represents the core of the design. *Model mix complexity* awards points according to the number of different platforms, body styles and models, all scaled by the number of different body shops and assembly lines in each plant. *Part complexity*[13] can also be part of the peripheral variety, for it is partially driven by consumer choice, based on the number of engine/transmission combinations, wire harnesses and colour paints. *Peripheral variety* has more to do with customer preferences and plant sequence planning. The two major classifications (fundamental and peripheral) are two most important points to consider from this paper.

However, product variety poses extra challenges to manufacturability, requiring high flexibility from a particular manufacturing firm to produce

[13] It differs from the notion of *Component complexity* established in this thesis.

economically viable products [3, 142]. Ishii *et al.* [143] and later on, Ishii and Martin [144, 145] studied this situation and proposed what they called "Design for Variety", as a tool to focus on methodologies which help companies to quantify the costs of product variety.

Because of the large amount of variables involved during the estimation of product variants and costs of pursuing this line of production, Ishii and Martin, proposed a set of design charts and indices to assess the complexity of design for variety [146]. Indicators such as:

a. ***Part/component commonality or commonality index (CI)*** – Indicating the number of unique parts, (CI must be greater than *nil* and less than or equal to *one*; CI = (0, 1]).

$$CI = 1 - \frac{u - \max p_j}{\sum_{j=1}^{v_n}(p_j - \max p_j}$$

Equation 4.4 - Commonality Index (Martin and Ishii, 1997)

Where: $0 < \text{CI} \leq 1$
u = Number of unique parts,
p_j = Number of parts in the j^{th}-model,
v_n = Final number of varieties offered.

b. ***Product differentiation or Differentiation index (DI)*** – Indicating product bifurcations and landmarks that might (or might not) add value to the product. (DI must be greater than *nil* and less than or equal to *one*; DI = (0, 1]).

$$DI = \frac{\sum_{i=1}^{n}(d_i \times v_i \times a_i)}{n \times d_{avg} \times v_n \times \sum_{i=1}^{n} a_i}$$

Equation 4.5 - Differentiation Index (Martin and Ishii, 1997)

Where: $0 < \text{DI} \leq 1$
v_i = Number of different products existing in the i^{th} -process.
n = Number of processes,
v_n = Final number of varieties offered,
d_i = Average throughput time from the i^{th}-process to sale,
d_{avg} = Average throughput time from beginning of production to sale,
a_i = Value added at the i^{th}-process.

c. ***Product switchover or Setup index (SI)*** – Indicating how substantial product setups are being considered. (SI must be greater than or equal to *nil* and less than or equal to *one*; SI= [0, 1]).

$$SI = \frac{\sum_{i=1}^{n}(v_i \times c_i)}{\sum_{j=1}^{v_n} C_j}$$

Equation 4.6 - Setup Index (Martin and Ishii, 1997)

Where: $0 \leq \mathrm{SI} \leq 1$
v_i = Number of different products existing in the i^{th}-process,
c_j = Cost of set-up at the i^{th}-process,
C_j = Total cost (material, labour, and overhead) of j^{th} product.

First, this set of metrics for product variety can be incorporated to the collection of metrics proposed in this thesis.

This chapter summarises some of the work done on the estimation of complexity in engineering design. One can see that, in terms of product evaluation, it has been studied in two specific areas: the complexity of conceptual design or concept generation (the design problem of) and assembly sequence generation. Nevertheless, there are still many issues that need to be addressed and understood in order to build a robust kit of metrics that evaluate the complexity of the product. The following chapters further elaborate on these notions and present new ways of estimating product complexity. In the end, this thesis proposes the creation and implementation of a point-to-point system of indicators, whose ultimate goal is to produce an optimum and efficient product, which supervise and monitor the evolution of the design.

Chapter 5

COMPLEXITY ASSESSMENT IN PRODUCT DEVELOPMENT FOR PROACTIVE DFA IMPLEMENTATION

From the previous chapters one can wonder whether a product can be analysed as a system. If so, what sort of system would it be? Can the methodologies, exposed in chapter 3, for complex systems be used to analyse the complexity of a product? Is a product a system or is it part of a system? These points need to be addressed and clarified before proceeding to extract information about how to identify the sources and causes of product complexity.

The first part of this chapter is dedicated to identifying the similarities and differences between products and systems. Subsequently, dealing with these issues leads to the identification of product complexity sources. These sources are consequently categorized in order to draw a set of metrics and indicators that can evaluate them. As a result, a scheme representing the taxonomy of product complexity is devised and proposed as a framework for tackling and managing the sources of such complexity.

5.1 Complex systems vs. complex products

Frequently, when assessing complexity, the terms "products" and "systems" are used indiscriminately by academics and practitioners alike. From the previous chapter, one can realize that products could be classified as *systems*, because they have 'hierarchical' arrangements which reflect the spatial aspect of a system. Referring once again to the work presented by Koestler [44] "... *all hierarchies have a 'part within part' character* ..." The term 'part' within engineering design is by far the most ambiguous term. A 'part' could be a 'component' or a 'sub-assembly' – that is a Holon, according to Koestler. Take for instance 'a car door' (with all the components within), which for some designers/manufacturers can be taken as 'one part', it can be also thought of as a 'sub-assembly', whereas 'a handle', from the same door, could be considered as 'one part' belonging to 'a door', hence taken as 'a component.'

Nonetheless, products bear some special traits that make them unsuitable for the definition of 'a system'. Systems (i.e. manufacturing) as well as products are purpose-built entities. Products, however, have a limited lifespan. Products depend on the current trend of the market. They are developed to solve a particular need and once that need changes, such a particular product will be discarded, either by replacing it with a newer version (of the same type of product) or simply by discarding it altogether. Products are therefore finite and "sealed off" entities (so to speak). On the other hand, systems have characteristics as noticeable as: *self-organisation, emergence* and *adaptability* (although, as will be seen in the next chapters, products have gained some flexibility with the introduction of modularity or recycling awareness in their designs). Therefore, systems are not so disposable. They are engineered not only to be of a much larger scale, but usually for a longer lifespan. Engineering systems have a remarkable characteristic: *dynamism*. Whether the systems considered are power grids, transportation systems, manufacturing systems or comparable sort of systems, they need to be above all: *flexible*. Systems are expandable, upgradeable and open ended. Products, on the other hand, are static[14] entities that once manufactured will remain unchanged throughout their entire existence. This statement is debatable. It can be argued that products once finished, if modified or upgraded, would turn into different products. It can also be said, that products such as 'a car' do not change at all if tyres or mirrors (accessories) are replaced or upgraded. Conversely, it would be an entirely different car if the engine (the core of the car) were changed. For that matter, systems are more tolerant.

[14] Perhaps the term 'static' is too strong, but it is a classification that somehow reflects the fact that products are closed entities. Product modularity has somewhat made this label less apt, for it touches on both the spatial/structural and functional characteristics of a product.

Nevertheless, a product, unlike a part, can be considered as the nucleus of the system. Products are the *raison d'être* of manufacturing/assembly systems in general. Systems, such as manufacturing systems, are constructed and designed to fabricate a specific set of products. So it does not matter much if a product *is* a system or it *belongs to* a system, in this case, what really matters is that the evaluation of products will raise an interesting question: *Is the product leading the complexity of the production system?* Or *is there a mutual constraint?* Surely there is a mutual constraint. Products can only be manufactured if the system is capable of doing so; however this making of a product pushes the system to be expanded, upgraded and made more agile. The reason for these questions, amongst others, is the need to estimate the time and effort required to manufacture the product, as well as the costs involved in building and planning a whole process around it.

5.2 Product metrics

Firstly, it is necessary to observe how the different types of complexity interact. It is also beneficial to identify the factors upon which they are dependent. Since there is nothing published that clearly supports this point of view, the following model is proposed as the foundation for the postulations herein expressed.

The foundation of this series of analyses lies on two basic assumptions presented below and depicted in Figure 5.1

- A large number of simple components generates a complex assembly and therefore a highly costly one.
- A small number of complex components also generates a complex assembly and once again is highly costly.

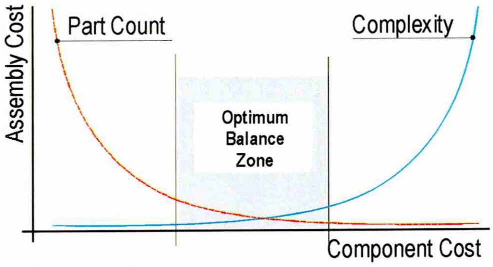


Figure 5.1 – Influence of component complexity and part count in product cost

Such exponential behaviour can be intuitively followed by considering the number of components. Without doubt this model raises the following interesting question: *when is a product simple enough, yet part-count efficient?* Design-for-Assembly (DFA) methodologies' main line of attack is part-count reduction, yet very often this sort of reduction in part count either results in a completely different product being constructed or brings along extra burdens to the manufacturability of the product components. Evaluation of the manufacturing processes is an important aspect of the DFA methodology. It is required for each component of the product and it is based upon a subjective shape complexity analysis, reminiscent of part coding. However, as parts are integrated or eliminated to optimise assembly with respect to part count and assembly operations, inevitably more complex components are created and more complex insertion processes are required. The overall assessment requires that the time and cost of production is minimised, but this obviously requires a compromise between manufacturing overheads, assembly processes and part count. Consequently, parts being combined or re-designed, with the intention of accommodating extra information or particular functional requirements, presents the designer with new challenges.

In order to put the whole analysis in perspective, the following sections will lead into the various considerations given to the product development process, and in doing so it describes reasons as to why and how complexity is to be analysed.

5.3 Complexity across several stages of development

The foundation of this work is the development of a product structure as a top-down activity, where the product architecture undergoes a series of improvements, modifications and refinements throughout its development timeline.

The development process requires the assessment of various types of complexities. It also requires the setting of a proper context for their evaluation, due to the context-based characteristic of complexity. This new setting immediately prompts for diverse set of complexity metrics to be obtained from and for the several stages of the product development process, hence defining a particular context for every single one of them (i.e. concept stage metrics, detailed design metrics and the rest). This practice

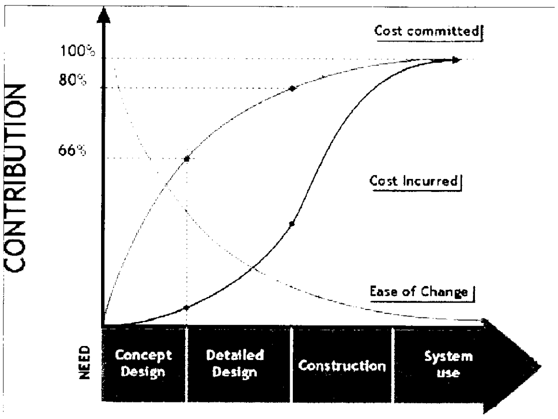


Figure 5.2 – Design changes vs. Product development cost

follows a 'timeline', in which the design processes have been associated with their ease (or rather lack of ease) of change (see Figure 5.2 – adapted from the Lucas DFA methodology [11]). It has been said that the product architecture can be easily changed at the beginning of this timeline, but hidden in this picture are the myriad of opportunities and variations the designer has to deal with. Therefore, it is not surprising that most designers choose to follow one line of reasoning or a specific design procedure that diminishes the number of possibilities and 'trial-and-error' cases.

Furthermore, the number of combinations in terms of *design parameters, functional requirements, user requirements* and *possible solutions* are the ones that outline the ever increasing '*sense of complexity.*' This situation can take place either during the creation of a new design or throughout the optimisation of another that introduces new technologies. However, it is possible to find oneself in this situation, whenever product complexity is considered only as a function of the number of choices available.

5.4 Knock-on effects in product development cycle

Every stage has its own complications, let alone the blurred distinction between any two of them (Figure 5.3). Whatever decisions the designer makes at one stage will affect the choices available at the next. Nevertheless, the further the development goes towards the right-hand side of this timeline; the less abstract the product becomes, but resources have been already committed to its creation, hence the decrease in the ease of change.

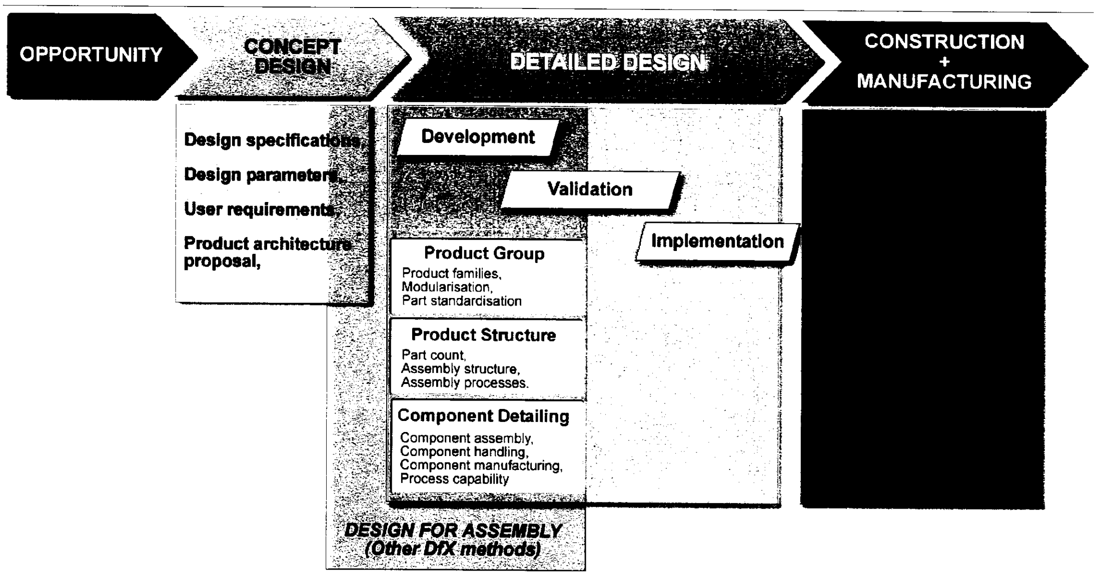


Figure 5.3 – Several sources of complexity across design stages

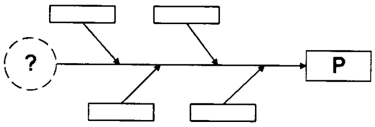


Figure 5.4 – Development process pipeline

This evolution in the product architecture can be seen following a pipeline of transformations. It begins with the initial 'spotting' of the opportunity and continues to finally producing a solid representation of these thoughts. As this 'pipeline' (Figure 5.4) transports the design intent, from those 'initial thoughts' to the 'final construction', every design stage will be adding its own complexity. This procedure sometimes obscures the design intent, further misleading the designer into achieving something that does not agree with what was originally proposed. Ideally, this situation could be avoided if those add-ons in terms of types of complexity are evaluated to see how they reflect on that of the final product, and preferably their influence controlled.

When designers start formalising their concepts, the product architecture is being defined in its interactions and functionalities. Even in a non-physical world, this initial scheme has a form. Consequently, this so called 'cloud' of interconnected ideas will be carried on to the next detailing stage, where a much refined process of formalisation or geometric definition will take place. The product then is enriched with information such as: groups, modularisation, standardisation, assembly structure, assembly processes, and others. Thereafter, the product configuration needs to be validated and finally implemented, ready for manufacture. The concept design stage has therefore stated the foundation upon which the 'development process' should start working.

Since this work is particularly interested in DFX methods, then the management of all the 'design requirements', 'parameters' and 'specifications' will not be considered as the starting point for the evaluation, but rather the first idea of how the product would look or, in other words, the 'initial product architecture' that has been proposed within the 'concept design' stage.

At the same time as the 'design intent' travels along the product development pipeline, as depicted in Figure 5.5, this pipeline gets extra layers of complexity added on to it. These layers are characteristic of every stage in the design process[15]. They will ultimately influence the final outcome, but it can be also thought of as distorting the view of the initial design intent, namely solving a problem or satisfying a necessity.

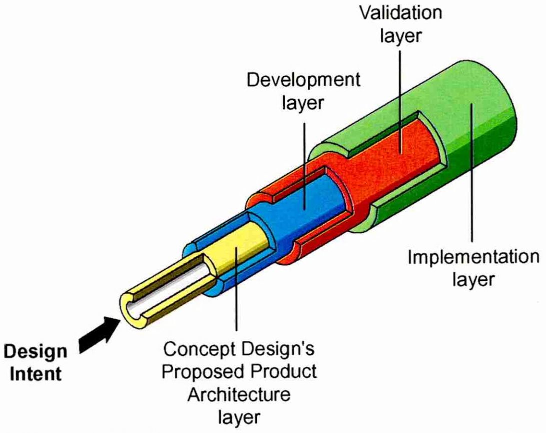


Figure 5.5 – Layers of complexity added to the product as it follows the design process

5.4.1 Non-linearity of complexity layers in the process

Unfortunately, the addition of layers of complexity to the development process pipeline cannot be taken as a straightforward and linear calculation of complexity. All the layers tend to influence each other, that is, there is not a basic proportionality between one source of complexity (cause) and another. This triggers one or multiple effects on the downstream process – that is if a concurrent approach is to be followed in the design. It follows that various types of complexity cannot be added or subtracted so easily. This, as well as how complexity is considered within the product, will be covered later on.

[15] It can be argued that some stages are not necessary at all, but since these represent the general idea of the design process as is considered today, then the representation is still valid. The idea of whether the design process is the right one is not considered at all in the scope of this work.

As mentioned above, the difficulties presented at every design stage are reflected in the final complexity of the creation of the product, and it is the designer that must decide how to manage the sources and the complexities produced. If the designer is to pay attention to this myriad of possibilities every time the product enters a different stage, he will be spending more time struggling with them, than with his main goal: the product. If, on the other hand, the designer were supported by a set of metrics that alert him of the risks incurred when increasing or decreasing the values of certain parameters, then he would be able to decide upon which line to follow, thus achieving an efficient design or at least making it more likely that he/she will.

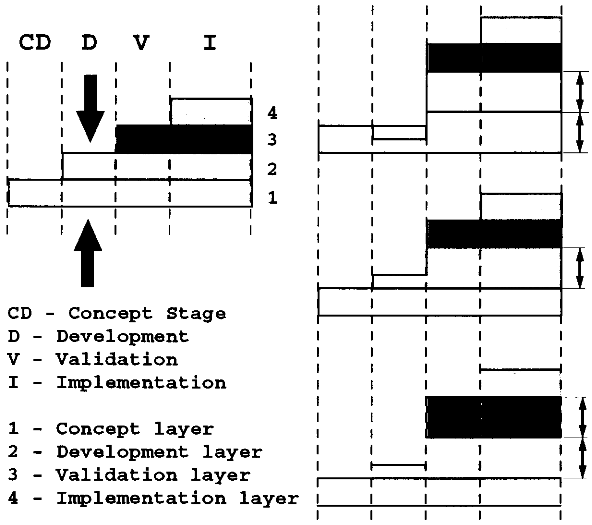


Figure 5.6 – Risks presented when adjusting the level of complexity at anyone stage

Figure 5.6 illustrates the amount of possible combinations that can be produced after adjusting some of the layers of complexity imposed[16] by the design stages. This somehow highlights the idea of complexity being constant. As expressed by many authors, the complexity of a system seems to be a characteristic of the system itself. In the area of physics this concept is presented in the work of Agudelo-Murguía and Alcalá-Rivero [21], where "Complexity" is defined as an inherent property of the system that evolves when a higher and more diversified number of elements interact amongst themselves. This position is shared by Tang and Salminen [147] in their view of complexity as a beneficial characteristic of the product as long as the product's complicatedness is reduced, this interesting perception is in part the foundation of most of the ideas presented in this work and it was further explained in chapter 3.

Another striking concept has emerged in the area of biology and evolution. There is no need to comment on the implications of this theory as it would be out of

[16] As seen later, the introduction of additional complexity is something that cannot be avoided and is part of the very nature of interaction between the product and the environment.

the scope of this work, but the concept is well-suited for the ideas considered in engineering. Evolutionists have been challenged by the notion of 'intelligent design' and more specifically by that of 'irreducible complexity' [17] in biochemical systems. The author Michael Behe has been the one leading this trend. His explanation of the term fits in perfectly with the idea of defining a way to get to the core of the complexity of a product:

> " ... By i*rreducible complexity* I mean a single system composed of several well-matched, interacting parts that contribute to a basic function, wherein the removal of any one of the parts causes the system to effectively cease functioning ... "

This is such a powerful statement that, by itself, it defines the core of the system. If it is viewed in the light of product complexity, it could be validated for every single stage of the development process. The goal is then to find the essence of the product, adding the minimum, but required, number of layers of intricacy to the above mention pipeline, which are generated at every stage. As simple as it sounds, this essentially has to take into account the view of the designer, because the designer is the one that makes the decisions of what, when and where to add in or take out parts/components in a single product, with the consequences that it brings. Since the designer is the only one that knows the requirements that the product is meant to satisfy, he and only he is the one that will manage all the data that are presented to him. The manner in which the data are presented is another factor, and it is precisely one of the novel ideas introduced in this work.

As the designer 'walks' along this development process, he encounters tools that would help him achieve his goal more effectively, namely DFX methods. These methods give much needed advice, but sadly as an aftermath situation, that is, when the designer has almost defined the look and architecture of his product. However, if these methods are applied proactively (i.e. as the product goes through this process), then the benefits of DFX will be highly appreciated. It poses the question as to why they are not implemented proactively at the moment, and some of the reasons might be:

[17] Introduced by Michael Behe in "Darwin's Black Box: The Biochemical Challenge to Evolution". 1998. Simon & Schuster Inc. 318 pp.

1. *Lack of understanding*: Product architecture is very abstract, and the designer has not formalised completely the design concepts,
2. *Lack of balancing*: Even if DFX methods cover the 'concept stage' and 'validation stage', some "improvements" at this stage do not reflect very well in others downstream, a very well-known example is that of reducing the number of parts to its minimum, then the analysis of every component will take longer.
3. Perhaps most of the factors involved have not been classified yet (*Lack of exploration?*)

This dynamic modification and interaction of parameters to satisfy the requirements of the actual development can be very frustrating if no assistance is provided (e.g. an appropriate presentation of helpful data). This assistance must allow the designer to carry on with his activities, which are making the most of his creativity and getting the work done on time. This is where the evaluation and estimation of complexity metrics for a proactive DFA implementation will come into play. Once some of the sources of complexity have been recognised, the next step would be to learn how to deal with them and finally make them work to meet the designer's own interests.

Nevertheless, in order to offer the designer helpful information, such information will have to be captured, assessed and fed back properly. In a recent article Thilmany [148] describes the state-of-the-art in information capture and storage. A concept for using databases as a mapping system to represent philosophical arguments or thought processes for what is called: the concept map. What is relevant to this work is the concept of recording engineers' thought processes via computer, as they design a part. Thilmany comments on the large amount of information generated during the design process (design rationale) that is not recorded. This important amount of information is lost as designers are detailing and refining their ideas towards the final stage: Construction of the product. The author effectively describes some techniques that have been tried to register such design rationale (e.g. construction of libraries of already solved problems as a way to find solutions to current problems in past solutions).

Recording design steps or registering historical design data can be also accomplished by measuring characteristic factors in the design process. These factors can later be mapped on to the assembly and/or manufacturing process. Such factors would help identify key issues that, as in this case, increase or decrease the complexity of the product. This would be a dynamic observation of the evolution of the product, where the metrics represent unique characteristics of the product structure, in other words, a *signature of the product*. (See section 5.6.3 for further development of this idea)

The extraction of metrics during the product development process can present other benefits, as highlighted by Hauser [149]. The author uses the common (but often overused) example of a 'thermostat' to present the need for the creation of an *adaptive control mechanism*. This mechanism must be able to deal with changes about an operating point, deriving practical and robust methods that consider the data available in the organisation. He makes his point by stressing the need for selecting the right metrics, establishing a culture to reward those metrics and teach that culture on how to use them properly. Some of the metrics proposed in different fields [56, 58, 59, 70, 72, 76, 95, 150-155], which deal with the product development process, have been already discussed in chapter 3. Once again Tang and Salminen [147] are amongst those people that spotted the need for *quantitative complexity metrics*.

5.4.2 Trade-offs amongst several DFX methods

Perhaps one of the biggest obstacles in achieving the 'perfect' product is the amount of requirements to be met. One product can score exceptionally well under a Design for Manufacturing methodology, but it can rank quite badly under a 'design for recycle' or a 'design for assembly' technique. Therefore, what would be the product that ranks best in this multi-dimensional, quasi-impossible-to-portray environment? Well sadly, it all depends. It depends on the options presented to the organisation that is creating the product, and this is when things start to present a much darker view, because the product is not only trying to please the several DFX methods, but trying to appeal to the organisation itself. So in this constant confrontation of DFX's something has to give and that is, probably and as expected, the weakest of the requirements.

5.4.3 Traceability and dynamic planning

Unquestionably, multiple trade-offs amongst the several DFX methods present a myriad of possibilities to get the product right, but more often than not to get it "wrong" the first time, hence the iterative design process. However, if concepts or components created, and thought of as part of design, were traced and mapped onto the most likely representation of the finished product, designers would surely be able to foresee the possible consequences of the modifications they introduced at every step.

In order to extend the above mentioned concept, the following scenario will be presented as an example. Suppose there is "Product A" which has a certain number of predefined components (maybe reusing, or not, already proven components from previous projects) or entirely new components, of which the designer knows nothing -in terms of shape or geometric appearance- other than their required interaction with neighbouring components. Suppose as well, that there is a monitoring system which would be recording every available piece of data such as number of components added (either physical or mere concepts), number of interactions, as well as the type of interactions required. This "monitoring system" would then perform a DFA/M analysis on the spot, with all available data and would immediately highlight possible faults in the design, that is, those situations discouraged by typical DFA guidelines.

With the aid of this monitoring system, designers could then try several product configurations without actually having a finished product, in other words, as components are added (once again, either as geometric shapes or simple concepts), designers will continually have a picture of the possible consequences of these actions. Designers would then be able to act on "what-if" trials, because they are continually supported in their decisions.

As depicted in Figure 5.7, systems that can monitor changes and trace them on a timeline, might be able to highlight the critical points at which decisions made by designers have increased/decreased the complexity of a product. These critical points or bifurcation points can be thought of as "milestones". Although the well-known adage "not making a decision is a decision in itself" still applies, it is also valid that a decision is made every time a modification is introduced in the product structure. The main idea is that these 'milestones' can show what modifications had a greater impact on the end product and the subsequent consequences downstream. Perhaps this could be another way of detecting relevant decisions and therefore being able to learn from mistakes?

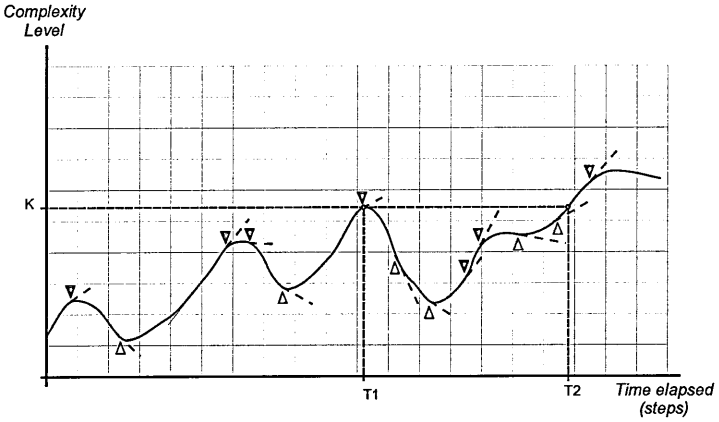


Figure 5.7 – Histogram. Complexity at every time step for risk evaluation (consequences)

This "histogram of design decisions" (see Figure 5.7) plots 'time[18] elapsed/steps' against 'complexity level.' Before going any further and for clarity's sake, something must be said about this level of complexity. This thesis emphasises the idea that there is not a single numerical value that can convey the idea of the whole complexity of a product. Therefore the fact that this graph shows such a single figure as being so representative seems to contradict the overall premise that "complexity measures cannot be boxed into a single numeric value." The complexity level shown in this graph would be mainly for visualisation purposes and the reasons why can be presented as follows.

Complexity metrics can be seen as a set of monitoring tools that indicate which constituents of the product present higher intricacy and, therefore, which elements are bound to draw in more resources and design effort. This set can be formed by metrics relating data such[19] as assembly sequence (Csq), modularity options (Cmd), and geometric-interface intricacy (Cgeom) of the components amongst others. Based on the assumption that these metrics will share some commonalities (as expressed in the following section) they may as well be summed

[18] Time elapsed does not necessarily have to be measured in time units (seconds or minutes), but in 'steps', that is, as the designer modifies the product, either by adding or deleting specific features, these modifications can be used as steps and hence as a measure of design progress through time.

[19] Proper names for the intended metrics are presented here and subsequent chapters, but these labels are presented for the sake of argument.

to produce an overall complexity index or "overall complexity level". Another more interesting option would be that of considering the set of metrics as belonging to a ***complexity tensor*** and its magnitude as being the 'overall complexity level' of a product. Therefore, one of two representations can be seen as in Equation 5.1

$$K_1 = \frac{w_1 \cdot Csq + w_2 \cdot Cmd + w_3 \cdot Cgeom + \ldots}{w_1 + w_2 + w_3 + \ldots}$$

Equation 5.1 – Linear combination of complexity metrics to produce the overall complexity level

This linear combination can be accomplished with or without the introduction of "weights" that emphasise the importance of every metric within the overall value of the complexity. Conversely, the second representation can be offered a scalar value obtained as in Equation 5.2.

$$K_2 = \sqrt{(Csq^2 + Cmd^2 + Cgeom^2} + \ldots)$$

Equation 5.2 – Scalar value of the complexity tensor to represent the overall complexity level

The latter representation seems to be more in accordance with the magnitude of a tensor and, consequently, more likely to be the representation of choice for the 'overall complexity level.'

Regardless of the representation selected, every 'overall complexity level' at each 'time step' ought to be mapped to a set of possible consequences, should the product remain with the current configuration. As mentioned above, this set of consequences can be envisioned as the result of applying DFX analysis on the spot.

As seen on the graph above (Figure 5.7), two different 'time steps' can be seen as producing the "same" overall complexity level. Does it mean that one can arrive at the same consequences later on in the design process? Certainly not, and this is the main reason why a single numerical value for the overall complexity assessment is discouraged!

It might be that for the steps 'T1' and 'T2', the overall complexity value be the same value of 'K'. However, it can be seen that the value of 'K' can be achieved by a combination of factors as represented in Equation 5.1 and Equation 5.2, hence, to extrapolate the consequences of making certain decisions of steps 'T1' and 'T2' one does not refer to the value of 'K', but to the factors that produced the value of

'K', for this reason it was said that the 'overall complexity level' depicted in Figure 5.7 was for visualisation purposes only.

It could be argued that this method of "predicting" the shape of the final architecture is flawed, for there are uncertainties within the future choices and decisions made by the designer before actually reaching that final stage of the product. Certainly that is a valid argument, but it can also be said that at least the designer is receiving some feedback as to what the product would look like once finished and how it would behave against the guidelines stated by DFX methods.

Most people, and surely most designers, have a mental plan of what they want to achieve once they embark on a particular task, however engineers and designers are constrained by the amount of resources at their disposal and the amount of time given to their project. Methodologies such as DFX offer certain guidelines that help to optimise the product, and these guidelines would give greater benefits if designers could use them for their planning.

As mentioned in chapter 3, the work presented by Van der Heijden [47] on strategic planning highlighted three schools of thought: the rationalists, the evolutionists and the processualists. Firstly, the rationalists, who plan to achieve a specific goal, no matter what path they follow, as long as they can reach that objective (with the drawbacks of uncertainties that will send the process off the track). Secondly, the 'evolutionists', who base their approach on reactive steps, taking into account the emerging improbabilities, but fail to act on time or quicker. Finally, the school presented by the 'processualists', who base their approach in a compromise between the two previous perspectives, planning ahead in small steps but reacting consistently depending on how the 'process' behaves and then, if needed, introducing modifications to actively influence the subsequent stages.

This issue is brought back again here due to its relationship with the idea of recording a histogram of complexity levels during the design process. As the product development takes place, designers can estimate but not determine whether their initial targets will be met. After a careful planning of how to use the resources at hand, designers will have at every step a reminder of the guidelines offered by the different DFX methodologies and thus fine tuning their progress. This practice reveals the true nature of dynamic planning and follows the idea presented by the "processualists" in strategic planning, as mentioned above.

Now that the reader has an idea of how these metrics of complexity can be put to good use, the following sections will present some ideas of how to extract the different metrics and how this should function.

5.5 Complexity Assessment – Ranking Theory

As Hauser [149] quite rightly suggested, the analogy of a 'thermostat' can be used as an example of how and why metrics are required during the product development process. In this analogy, the development process is presented as an iterative practice which seeks to maximise profits even if the environment is changing. Whatever the type of metrics used – customer satisfaction, number of components, assembly sequence suitability, and time to market, amongst others- they all provide the same information and feedback routine to fine-tune the performance of a firm, organisation or design team.

In a recent conference, Eckert *et al.* [156] presented a paper in which they warn designers of the dangers of using quantitative metrics in design research. Although the paper is directed towards the analysis of the objectives of research into design and its dealings with it as a complex human activity, the authors made a valid point in indicating some of the parameters to be met for any type of quantitative metrics. These parameters are: *Validity*, *Reliability*, and *Generality*, amongst others.

The rationale behind these three parameters can be best understood if one follows the questions raised by the authors,

- "*Validity – Does the metric correspond closely to the aspect of the phenomenon one wishes to measure?*" – This follows certain aspects of common sense, for instance, if one is to measure the suitability of two different assembly sequences, it is expected that the one most suitable will also appear to be more efficient in time, in other words, what one can perceive as "more suitable" can also be reflected by a metric that actually gives a higher score to the sequence selected.

- "*Reliability – Is the metric sufficiently stable to be a useful measurement of anything?*" Once again, this follows the general notion of a metric that can produce, within a certain accepted range of failure, a consistent measure of the phenomenon one wants to evaluate.
- "*Generality – How general is the relationship between the value of a metric and the factors that determine it?*" This principle of generality is what actually dictates the invariability of a metrics and its non-dependency on particular factors.

Above all, the assessment of the complexity of a product structure or design option will have to follow more basic numerical principles, such as: Uniqueness, Minimum or Zero level.

- *Minimum levels of complexity or Zero complexity* – The baseline of the scale in measuring complexity must follow some rather interesting patterns, it must not be able to reach the 'zero level', for a 'zero level' represents a product without components. As stated in previous chapters and at the beginning of this one, a product starts to present levels of complexity if at least two components make it. In other words, if a product has a least two components, these two will interact and it does not matter in what form, for this form of interaction will present unique level that separates the product from another with the same two components. Putting it even more simply, once two components within a product structure start to interact, product complexity is present, whether this interaction is the simplest form, depends on the possibilities that exist for these two components to interact, therefore the minimum level of complexity will follow the same definition as that given to "irreducible complexity" in section 5.4.1.
- *Uniqueness* – Due to the very nature of the concept of complexity and its embedded subjectivity, measures of complexity must be presented for different product areas. This notion will be further developed in chapter 9 ("*Product complexity across domains.*")

Finally, the metrics proposed in this thesis have been devised under the following premises:

1. Metrics of complexity will provide the means to encourage a proactive DFA environment. In other words, the targets set for an optimal, and therefore, less complex product are those that closely follow the guidelines suggested by the DFA guidelines for an optimal product structure. Moreover, these guidelines must be integrated into an assembly-oriented design environment, where a holistic approach to the design of the product is supported.
2. The monitoring, assessing and prediction of complexity, as presented in this thesis, will fall within the conceptual and detail design stages of the product development cycle. It, nevertheless, encourages the production of and integration with other metrics to fully observe the product development process and to keep this process within the range of optimal practices.

It is understood that, in order to provide a set of metrics that can be reliable, valid and general, these metrics must be as objective as possible and, therefore, any involvement with the user of the metrics will be kept to a minimum. The evaluation of complexity requires the analysis of multiple variables that imply higher production time, and hence higher production costs.

5.5.1 Spatial and functional complexity

Undoubtedly, functional and architectural aspects of the product act as independent characteristics yet are reliant on each other. As can be seen from the literature review in previous chapters, a product needs to be taken as a whole to understand it. This holistic approach means that parts cannot be considered individually, because they all bring into being the purpose of the product, when they act together. However, techniques such as DFA/M are directed towards physical aspects of the product, that is, its 'architectural characteristics', with the functional side in mind **(functional reasoning)**. Tuttle [114] quite rightly expressed his concerns about the dangers of performing DFA/M analysis without understanding the function of the product and the functionality of the parts within.

Consequently, there are plenty of examples of architectural analysis in the field of constraint-based modelling which have touched on the notion of related functionality. For instance, Kim and Lee [157] created an assembly-modelling environment such that the designer creates a design which may be readily analysed in terms of its dynamic and kinematic performance. Notably, Rajan *et al.* [158] considered the types of joints required to achieve certain functional requirements. Anantha *et al.* [159] used the absence of constraints to generate kinematic joint types in an under-constrained assembly model.

Product functionality is, by definition, outlined in the conceptual design stages of the product development cycle. Even if this research acknowledges the fact that functions of individual parts, more often than not, appear to have little to do with the function of the product (emergent behaviour, as described in chapter 3), this work is mainly directed at architectural analysis. It appears a more tangible and less abstract concept. It will be seen, however, that product complexity metrics must be used in conjunction, as there is not one single metric that encompasses all that is perceived as complexity in the product.

5.5.2 Setting boundaries

As frequently repeated, modelling and/or analysing complexity needs to be placed in context. This context may as well be an environment centred on the product being designed. It is surrounded by aspects such as: *tooling, handling, manufacturing processes* and *other products* (product families, product variety and modularity), to name but a few of the things that have the strongest influence of the product's architecture. Notice that it defines the product in terms of other products, which can be seen as an unnecessary tautology, but it shows that products are not isolated entities.

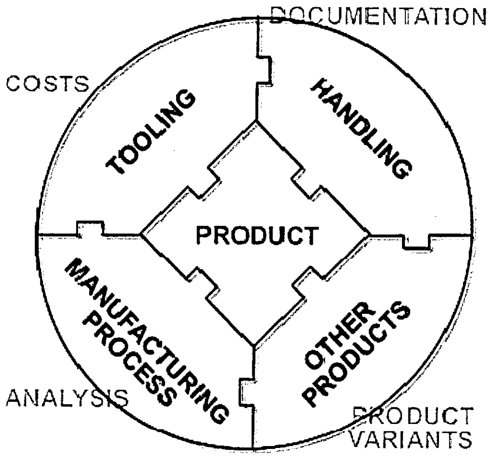


Figure 5.8 – Environmental constraints and feedback in product development.

These boundaries are based on a spatial and/or architectural-oriented assessment complexity as opposed to spatial and functional one. Since this analysis is devoted to the implementation of DFX techniques, architectural complexity assessment is, therefore, the core of this thesis. It, however, discretely involves functional aspects of the product, now that product functionality is the other significant subdivision of product complexity.

Working from the inside-out of this product-centre environment (Figure 5.1), it can be seen that the product interacts and therefore receives indirect influence from aspects as important as: *costs* (material costs, tooling costs, manufacturing costs, etc.), *documentation* (regulations, customer requirements, drawings, product representation, etc.), *analysis* (product validation, tests and verification, amongst others) and *product variants* (product families, modularisation, forward and backward compatibility, etc.).

It is clear that this representation is far too abstract and it is intended to be that way. As one moves from the outer to the inner layers or levels there is a narrowing of focus, which is another abstract classification on its own right. But it places the product in a dynamic context of interaction with its own manufacturing environment.

5.6 Complexity taxonomy towards Proactive DFA: (Levels of abstraction)

Much has already been explained about the building of this sort of classification in chapter 3. Nevertheless, a quick follows, to help give explanation to the development of the metrics produced.

As illustrated in Figure 5.9, the optimum balance for a product design between manufacturing capabilities and assembly operations can be seen from two different angles, namely, the concept '*component vs. assembly*' and that of '*static vs. dynamic*' complexity.

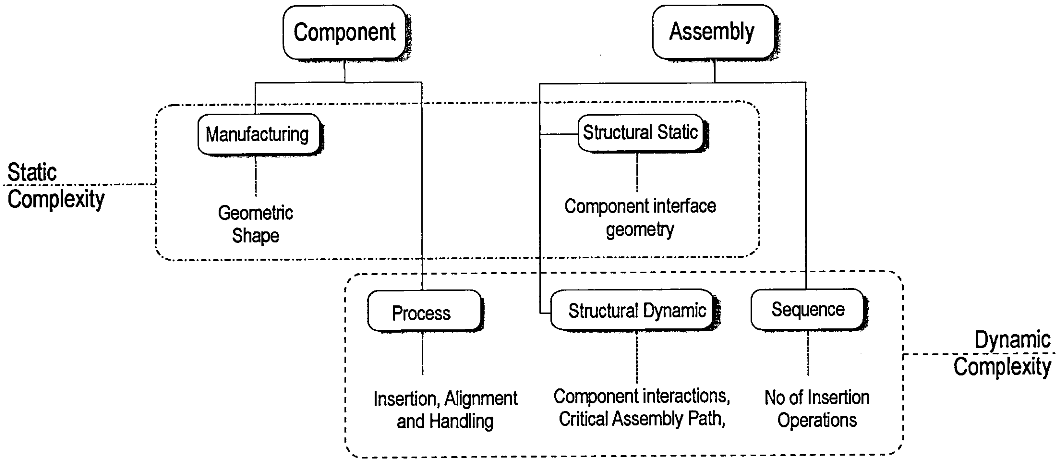

Component
Assembly
Manufacturing
Structural Static
Static Complexity
Geometric Shape
Component interface geometry
Process
Structural Dynamic
Sequence
Dynamic Complexity
Insertion, Alignment and Handling
Component interactions, Critical Assembly Path,
No of Insertion Operations

Figure 5.9 – Complexity analysis defined in terms of Component vs. Assembly and Static vs. Dynamic

Both are equally valid, but each view addresses the complexity evaluation issue from a slightly different perspective. The '*component vs. assembly*' model makes the interpretation of complexity a much more solid concept, where logical observations form the foundation of such a model, making the understanding of the whole idea easier to assimilate. The '*static vs. dynamic*' view, on the other hand, appeals as a much more abstract notion altogether, but it is perhaps this intangible condition that makes it more useful for a straightforward mathematical representation of the complexity evaluation methodology.

Whatever the model used to express the complexity concept, the relationship is fully interlinked meaning that the '*component complexity*' is extremely influential on that of the assembly and vice versa.

5.6.1 Sources of complexity during concept and detailed design

Not surprisingly, in order to extract meaningful information from the product and be able to build a model that can represent a complexity measure, a list of parameters must be drawn up. This list of characteristics which, mostly, affect the product complexity will be classified along the lines of the '*static vs. dynamic*' complexity taxonomy mentioned in the previous section. They will serve as a measure of classification amongst the several product structures that can be created during the design process and perhaps give a good advice to the designer as to which design would be the most suitable, in terms of easiness (*assemblability*) and practicality (*manufacturability*). For that reason Table 5.1 presents an inventory of constraints, classified either as static or dynamic, which will give a clearer idea of the principles they were based on.

Table 5.1 – General aspects considered for complexity evaluation

Complexity Aspect	Type	Complexity Aspect	Type
Adjustments	*Dynamic*	Feeding methods	*Dynamic*
Assembly technique - Automatic	*Dynamic*	Functional assessment	*Dynamic*
Assembly technique - Manual	*Static*	Geometric constraints	*Dynamic*
Assembly Sequence	*Dynamic*	Geometry – Section	*Static*
Component material	*Static*	Geometry – Tolerances	*Both*
Detectable assembly base part	*Dynamic*	Gripping analysis	*Dynamic*
Different material constraint	*Both*	Integration	*Dynamic*
Environment	*Dynamic*	Manufacturability	*Static*
Ergonomics	*Static*	Manufacturing cycle time	*Static*
Handling		Manufacturing process steps	*Static*
Handling difficulties		Material waste (near net shape)	*Static*
Abrasiveness	*Dynamic*	Mechanical tooling	*Dynamic*
Adherence	*Dynamic*	Modularity	*Both*
Flexibility	*Dynamic*	Number and type of surfaces	*Static*
Fragility	*Both*	Nr. of component interactions	*Both*
Gripping problem	*Both*	Number of components	*Both*
Need mechanical assistance	*Both*	Number of different parts	*Dynamic*
Need optical magnification	*Static*	Number of edges	*Static*
Need using grasping tools	*Both*	Number of faces	*Static*
Need two hands	*Static*	Number of features	*Both*
Nesting	*Both*	Number of re-orientations	*Dynamic*
Sharpness	*Both*	Number of Turnovers	*Dynamic*
Tangling	*Both*	Personnel skills	*Dynamic*
Untouchable	*Both*	Product Safety	*Static*
Orientation of the part		Product Service	*Static*
Rotational symmetry	*Static*	Production quantity	*Static*
Symmetry	*Static*	Relative movement	*Both*
Size and weight of part		Reliability	*Both*
Handling convenience	*Dynamic*	Risk (Nr. possibilities * Severity)	*Both*
Size	*Static*	Tortuosity	*Dynamic*

As expected, this list is by no means complete as it might require extra filtering. Some factors may as well be included or considered as part of more general variables, i.e. '*Number of edges and faces*' can be embedded into '*Number and type of features variable*'.

It can be assumed that the behaviour of any complexity-vs.-number of parts graph will be product specific[20], since it would be impractical to compare products across unrelated lines of application (e.g. domestic drill vs. oil exploration drill). However, such a graph will eventually produce the necessary estimations to assess the final product feasibility.

Chapters 5 and 6 look at those highlighted variables. Those variables will be arranged to meet the assembly and component needs in the best possible way. Additionally, these variables will help to develop a mathematical model that will allow the measurement of product complexity.

5.6.2 Complexity Indicators – Conditions and diagnostics

A minimum number of conditions must be met to produce a real indicator of product complexity. Whatever the evaluator produced, these conditions must guarantee that the information given is realistic and as accurate as possible, as well as helping to avoid mathematical or logical mistakes that might crop up during the implementation of such indicators in a computer. A set of the most basic conditions is presented as follows.

1. *Value of indicator must be equal to or greater than one* – as mentioned in section 5.4.1, all products are said to have a minimum level of complexity (irreducible complexity). Such level represents the basis upon which the entire structure of the product will be built.

2. *Value of indicator must be Positive* – following the first rule, numerical values reflecting levels of complexity in a product must always be positive, so as to meet minimum logical requirements. Consequently, these values can follow two basic arithmetical operations: addition and subtraction. As long as these do not violate condition number one.

3. *An indicator should not be just a numeric value* – It would not be fair to reduce a design to just a figure, besides, what information can be extracted from a single

[20] Product specific complexity metrics will be mentioned in chapter 8.

numeric value? Numerical values might be necessary to report on specific aspects of the design, giving an overall view of the complexity of the product, but they must trigger a message or advising legend, rather than presenting a simple number.

4. *There must be more than one indicator in action* – Analysing the complexity of a product is not just counting parts. It is also the evaluation of shapes, interactions, dependencies and so forth. Therefore there must be a series of indicators, monitoring sensors that is, which supervise continually the creation of every different product structure. At least, there must be one indicator per variable considered, so that when used in combination, they will give a realistic report of the product

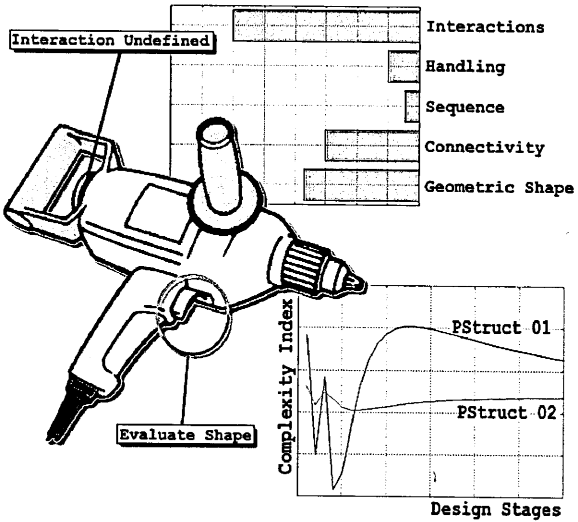


Figure 5.10 – Complexity Indicators – Monitors

For sure other extra conditions must be specified, not only to give a good feedback to the user – tracking and reporting on sources of a potential increase in the complexity levels, but to keep the product being design well balanced[21] and under control[22]. It ought to give continuous advice to the designer during the evolution of the product as well as helping to spot stages at which the product increased its complexity level; as roughly illustrated in Figure 5.10.

[21] Product balance is the trade-off between part-count reduction and component shape complexity.
[22] The use of complexity monitors must not constraint the designer, restraining his/her creativity.

5.6.3 Complexity indicators – product signature

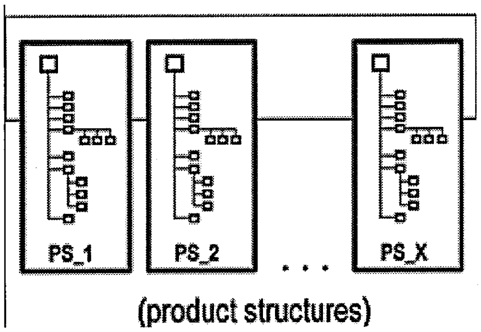


Figure 5.11 – Product structures to be compared

As presented in section 5.6, several complexity metrics must be used in conjunction to produce a real complexity indicator. The evaluation of complexity must start with the indications of assembly complexity as a whole and then step up in detail by calculating the complexity of its individual components. It is worth mentioning that much information can be extracted at early stages, even when the product is not fully detailed (e.g. geometry and material, amongst others). This helps to offer advice to the designer as early as possible. Logically, as the product is continuously refined, a much more accurate measure can be produced.

In order to compare two product structures, they must be at least at the same stage. It is understood that both structures must at least fulfil the functional requirements expected within that specific product. This is because the indicator of complexity of a particular design depends on the structure given. A basic example can be useful to clarify this point. Suppose for a moment, that there is a product structure PS1 expected to perform a "cutting action," then all components within that structure PS1 must interact in such a way that the task in question can be accomplished. If a second product structure, PS2, is produced to perform the same "cutting action," then these two structures have reached the same level of development and can therefore be compared.

As stated in section 5.4.3, dynamic changes can be tracked down within the product structure; it can also be seen when and which part caused the overall complexity to increase its value. If two different product structures are produced in parallel, it can be seen when these two start to differ and when one becomes less complex. It is as if every step were automatically analysed and recorded for future reference. Furthermore, if two structural configurations of the same product structure were formulated in parallel, then both arrangements could register their own complexity histogram.

The next two chapters will address the evaluation of complexity for assemblies and components, respectively. In connection with this section, it will be seen that the evaluation of assembly complexity using a matrix-based representation for the product structure will make possible the identification of local complexities, that is, which part of the product, from the point of view of assembly interaction, will present higher part-to-part interaction and therefore higher complexity values.

Chapter 6

ASSEMBLY COMPLEXITY ASSESSMENT

Assembly-oriented design requires the designer to tackle the task of creating a product from its composition and construction point of view. The fact that all parts must be considered at the same time certainly seems a daunting task and possibly an exhausting one too. However, breaking down the process into various stages can facilitate the implementation and introduction of indicators (or sensors) that monitor and trace the development process at any time. This will eventually reduce the sensation[23] of complexity attached to it. Once the product configuration has been outlined and/or formalised (later stages of the concept design phase), one could known whether the elected assembly sequence has been favoured at the expense of making the handling of components more cumbersome. Additionally, these indicators could point out whether components are so densely interconnected, that special tools might be required for their assembly.

The objectives of the Designers' Sandpit project (see chapter 2) require an integrative approach to Proactive DFA implementation, where operations dedicated to "Product Structure Support" involve assembly complexity assessment. Such an assessment focuses on dissecting the product structure itself, where by tracking component-to-component interactions and their interaction type (also referred to as 'interfaces') an estimation of the architectural complexity of the product can be given. This evaluation also exploits data already available in the product structure, namely, *component redundancy* (or repeated components), *component types, component clusters* (useful to offer support during assembly planning – e.g. parallel

[23] As mentioned in chapter 4 – the complexity perceived by the designer can be imaginary due to a poor understanding of the product.

assembly) and, ultimately, *assembly sequence complexity*, which has been thoroughly explored (see section 6.3.1). As a result, this information can be integrated into the assessment framework proposed in this work.

In this chapter, analyses of assembly complexity begin with the structural or architectural examination of the product (pair-wise component interactions) and the so-called 'structural periodicity'. These two topics constitute the least explored area in the product development process: its architectural composition. The examination finishes with a discussion of existing methods to address assembly sequence complexity. The concept of structural complexity evaluation is complemented with a collection of case studies presented in Appendix B.

6.1 Structural Complexity (Pair-wise interactions)

Structural complexity refers to the architecture of the product. Incidentally, this architecture is particularly sensitive to the way in which its constituents interact with each other. It is not surprising then, that in order to estimate the level of complexity, the quantity and type of interactions are two extra variables to be addressed, besides the number of components, that is.

In a product architecture or assembly, the number of components alone does not represent much, unless there is some sort of interaction amongst them. On the other hand, the amount of interactions per se, as a numerical value, does not say much either; the interesting issue about the interactions is their type. Assemblies can to be regarded as complex entities when interactions amongst their components are many and various.

Think of Figure 6.1 as the initial way in which components are expected to interact. They are somehow related to each other in terms of space, energy transfer, and data exchange, amongst other things. Taking into account their spatial proximity alone, much can be learnt from their ease of assembly, when their interactions are defined.

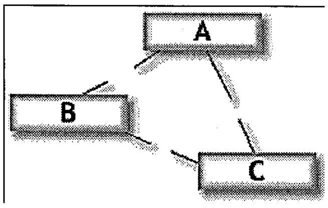


Figure 6.1 – Components interacting in an assembly

Components interact in several different manners or types, i.e. geometric matching or just a simple contact. Consequently, discriminating between the various types will give yet more information about the intricacy of the assembly itself. Type differentiation gives one of the first approaches to distinguish simple from complex assemblies, that is, assemblies whose components present just one type of interaction come out as simpler than those whose components present two or more types.

Figure 6.2 presents two examples of components interacting with different types of interfaces. In a structure such as that of Figure 6.2.a components connect to each other in a single monotone style (flat interface), whereas components in Figure 6.2.b relate to each other with a convex-concave interface (circular and square). This way, structure B, represents a much more complex assembly.

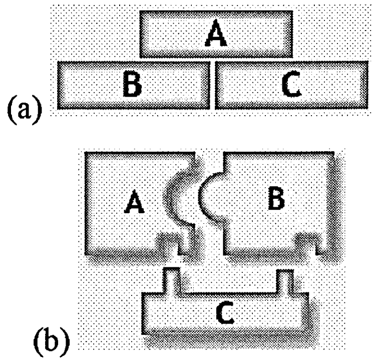


Figure 6.2 – (a) Single interaction vs. (b) Various types of component interactions

A first prototype of a mathematical model can be built following the initial conclusions presented above. In Figure 6.2.b, there are three components (n = 3), with two interactions each (l = 2), but two different types of interactions (w = 2) – these will be referred to as *interfaces*, see Section 6.1.2.5. In the meantime, Table 6.1 shows this evaluation in detail.

Table 6.1 – Evaluation of components interactions (example)

Components (P)	*Interactions (l) per component*	*Type of interactions (w) per component*	*Sum of Interactions*
A	2	2	$w_1 + w_2$
B	2	2	$w_1 + w_2$
C	2	1	$w_2 + w_2$
Components, N = 3	Per assembly, AL = 3		$2w_1 + 4w_2$

For an assembly *A*, with *n* components *P*, where $A = \{P_i\}_{i = 1, 2 \ldots n}$. and a number of interactions *l* per components; the index of assembly complexity can be illustrated as in Equation 6.1.

$$ACI_L = \frac{1}{2}\sum_{i}^{n}\sum_{j}^{l} w_{ij}$$

Equation 6.1 – Assembly Complexity Index due to component interactions – First approach

Where:

ACI – Assembly Complexity Index,
n – Number of components
l – Number of interactions per component,
w – Weighted value for every component interaction

The fact that the ACI is divided by two is due to the interactions being counted twice, once from every component point of view. If two components are in contact and the surface of contact is continuous, they are said to form a 'regular or continuous' interface; if the interacting surface is interrupted the interface is hence an 'irregular or disrupted' interface. From now on, all continuous interfaces will be referred to as 'interfaces' only, unless otherwise specified.

Every interface must be counted separately, as it might be possible that two components could share two different types of interfaces, that is to say, the components are in contact with each other at two or more different points – irregular interface.

Numerical values for every weight *w* can be extracted according to the classification of interfaces. These values can be combined (or added) to produce a more elaborated representation of such interaction, for instance, they can have geometric matching and relative movement at the same time, that is, $w_i = w_{geometric} + w_{relative}$. Actual names and values for different classes of interfaces will be explained in section 6.1.2.5 below.

6.1.1 Average Structural Complexity Index for Interactions

According to Equation 6.1, assembly complexity is the combination of all single component complexities, at least those complexities related to their interactions. Nevertheless, this might lead to wrong interpretations.

Suppose *Assembly I* has ten (10) elements, interacting within a single interface mode w_1. Every component is allowed to have binary connections (two interactions per element). With this information and Equation 6.1, it results in Equation 6.2-a.

a. $ACM_I = (L_1 + L_2 + L_3 \ldots + L_{10})/2 = (2{*}w_1 + 2{*}w_1 + \ldots + 2{*}w_1)/2 = 10{*}w_1$

b. $ACM_{II} = (L_1 + L_2 + \ldots + L_5)/2 = (4{*}w_1 + 4{*}w_1 + 4{*}w_1 + 4{*}w_1 + 4{*}w_1)/2 = 10{*}w_1$

Equation 6.2 – Variation in the number of elements

Now, for a different *Assembly II* with five (5) elements and four (4) interactions per component, and again with the same type of interface w_1, the same equation (Equation 6.1) yields Equation 6.2-b.

From Equation 6.2, it can be said that both assemblies are equally complex. Nonetheless, intuitively *Assembly I* sounds more likely to be 'easier' to assemble. Despite its larger amount of components, its interactions are twice per component. The intricacy presented in *Assembly II* is down to the several interfaces presented, regardless of their being of the same type w_1 in both assemblies.

In order to make these results reflect a more realistic finding, it is possible to even out the index obtained. This index is divided by the total number of components involved. This 'average' complexity will indicate how interlaced an assembly really is. Applying this inference to the previous example yields Equation 6.3 (a. and b.).

a. 'Average' $ACM_I = ACM_I / n\ (= 10) = 2 * w_1$, and

b. 'Average' $ACM_{II} = 4 * w_1$.

Equation 6.3 – Average adjustment due to the number of elements

Now, Equation 6.1 can be modified to show this new finding, and be transformed into Equation 6.4.

$$ACI_L = \frac{1}{2n}\sum_{i}^{n}\sum_{j}^{l} w_{ij}$$

Equation 6.4 – Assembly Complexity Index due to component interactions – Second approach

<u>Example</u>: A numerical example will clarify what has been just explained. Suppose once again an assembly with three components A = {P₁, P₂, P₃}, every component is believed to have binary connections. Interfaces in every interaction are all different, that is, the set of interactions is presented as L = {w₁, w₂, w₃}. This information, summarised in Table 6.2, is valid to apply Equation 6.4, where the Assembly Complexity Index (ACI) will be ACI = 0.88/2. For the sake of argument, arbitrary values were selected for w₁, w₂, w₃, but selection of numerical values will be explained in section 6.1.4.

Table 6.2 – Components interaction - example

Component ID	*Interactions per component*	*Sum (for arbitrary values) w1 = 0.85, w2 = 0.32,w3 = 0.15*	*Numerical Value*
P_1	2	$w_1 + w_3$	1.00
P_2	2	$w_1 + w_2$	1.17
P_3	2	$w_2 + w_3$	0.47
		$2^*(w_1 + w_2 + w_3)$	

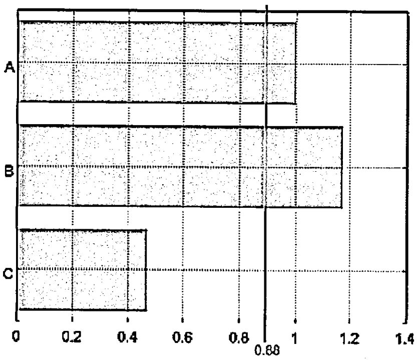


Figure 6.3 – Component interaction complexity.

Figure 6.3 illustrates the values obtained in the previous table. The dotted line shows the ACI value. It follows that; the complexity of an assembly cannot be higher than the highest level of complexity presented by a component with the maximum number and variety of interactions. Nor can it be lower than the minimum level of complexity introduced by a component with the least number and variety of interactions. ACI represents 'assemblability', not the complexity of the whole product. A holistic product complexity evaluation captures its 'assemblability', plus the complexity of every particular component.

The complexity of a single component alone does not affect the 'assemblability' of the product in terms of connections, but it will certainly have an impact in the sequence of the assembly, due to its shape and possibilities of being manipulated / handled. This position will eventually become clearer as the discussion evolves.

6.1.2 Component connectivity information in an assembly

6.1.2.1 Level of connectivity and deviation from minimum level

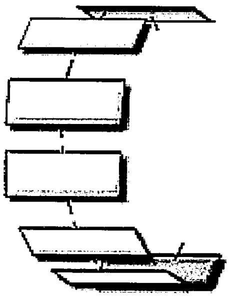


Figure 6.4 – Simple chain of components

Imagine a basic chain of interactions, where components are adjacent to each other and their interactions are down to 'previous' and 'next' as illustrated in Figure 6.4. If this composition is regarded as the most simple (**level one**), then any other configuration with a different level of interactions would have a 'higher' level of connectivity. The level of connectivity can also be thought of as the average number of interactions per component.

As a matter of fact, it follows one of the basic rules used by DFA methodologies: *Arrays that assemble vertically are the preferred option because of their low assembly time.*

This level of intricacy or interdependency indicates how much extra tooling could be required at a certain time for assembling all the components together.

Such level can be expressed mathematically as follows: The level of connectivity for an assembly A = $\{P_1, P_2 \ldots P_n\}$, with a total number of interactions $L = \Sigma\, l_{n.}$, is defined as the ratio between the total number of interactions L accounted from every component and the total number of components.

$$LC = \frac{L}{n}$$

Equation 6.5 – Level of connectivity in an assembly

Where:

LC – Level of connectivity,
L – Number of total interactions,
n – Number of components,

A basic example can be taken from the components in Figure 6.18. The total number of components within the assembly is ten (n = 10), and the number of links is thirty four (L = 34). Hence, the level of connectivity is $LC = 3.4$.

Notice that the amount of interactions (L = 34), represents the amount of interactions counted for every component and added together. This might sound confusing, as the number of total interactions seen in any given assembly is half of the number given by L.

Subsequently, the deviation from the minimum level of connectivity mLC (equal to two) for an assembly A, is defined as the *difference between the level of connectivity of that assembly (LC) and the minimum level of connectivity (mLC) for any assembly, divided by mLC (therefore, it can be expressed as a percentage)*. In mathematical terms is illustrated as follows.

$$deviation = \frac{(LC - mLC)}{mLC} * 100$$

Equation 6.6 – Deviation for any given assembly from the minimum level of connectivity (deviation from vertical assembly)

The value given by Equation 6.6 and represented as a percentage is a relative estimation of how solid or compact the assembly is. This is where the trade-off between part-count reduction and component shape complexity takes place. The making of more complex components implies that every component takes on more tasks to perform than those initially intended in the original design. That is, every component that has been re-designed following DFA rulings will have to handle a larger number of interactions with its neighbouring components. This value can also be read as "*deviation from simple vertical assembly*". It has been found that assemblies that have components that at least connect to two other components can be assembled vertically –there are obvious exceptions, but it is a heuristic approach.

Following the previous example and applying Equation 6.6 with an assembly for which the cohesion level value is 3.4, the deviation will be estimated as:

$$deviation = \frac{(3.4 - 2.0)}{2.0} * 100 = 70\%$$

Is it good or bad? A deviation higher than 25% suggests that additional operations are present (e.g. turning, simultaneous insertions or lateral insertions). Actually a level of connectivity of 2.5 or greater suggests that additional operations start to take place. Deviation from "simple vertical assembly" (that is, the minimum level of connectivity) can be thought as an early warning. As mentioned before complexity metrics cannot be read as an isolated figure.

Consequently, a level of compactness or cohesion can be extracted as well. This somewhat symbolises the effects of component modifications within the assembly. If components are modified, these modifications will have an impact on their surrounding elements. The level of cohesion is expressed as ratio between the number of actual links (L) and the maximum theoretical number of possible connections (as expressed in the following section, the maximum number of possible connections has been given the name of Upper Bound of the amount of component interactions 'UB'). Expressed mathematically,

$$CC = \frac{L}{n \cdot (n-1)/2} = \frac{links}{n \cdot (n-1)}$$

Equation 6.7 – Level of compactness or cohesion of the product architecture

Where

CC - Level of compactness or cohesion.
L - Number of interactions (adding up all components interactions),
links - Total number of interactions
n - Number of components.

It is worth mentioning that as in Equation 6.5, 'L' represents the number of total interactions, but this value was the sum of all the interactions accounted for every component, which inevitably meant counting the number of interactions twice. Hence 'links' is equal to half of 'L'

For an assembly with a DFA efficiency of 100%, it has been found that the level of cohesion is 50%, that is, it behaved like a perfect chain of components (*see Figure 6.4 and Appendix B: Re-design pressure valve*).

6.1.2.2 Boundaries for amount of interactions and variables held

Another type of metric covers the number of interactions that should be dealt with during the process of identifying component interactions and the amount of variables to be stored, which must track all the relations in any given assembly. As mentioned in section 6.1.2.1 above, a chain of components with a binary set of links for every component, can be regarded as one with the minimum number of interactions that can be given in any assembly.

Now with this minimum number of interactions as a basis, finding a maximum will set up an area in which all assemblies can be plotted and a ratio of 'connection efficiency' marked. This so-called 'maximum number of interactions' is found by determining the sum of the number of links that every component would have if it were adjacent to every other component.

Putting it all together in mathematical language, the minimum (lower bound) and maximum (upper bound) number of interactions can be calculated as follows:

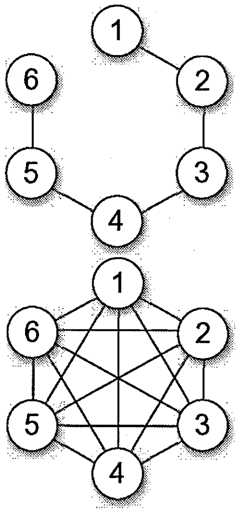


Figure 6.5 – (*Top*) Lower and (*Bottom*) Upper bound interaction graph

1. *Lower bound of the amount of component interactions* – For any given assembly *A*, with a total of *n* components P, the set A = $\{P_1, P_2, \ldots P_n\}$ is said to have a lower bound equals to *n – 1*, if and only if there are two links per component except for the first and last of the components in a chainlike arrangement.
2. *Upper bound of the amount of component interactions* – For any given assembly A, with a total of n components P, the set A = $\{P_1, P_2, \ldots P_n\}$ is said to have an upper bound equals to *n (n - 1)/2*, if and only if every component is linked to every other component in the whole assembly.

Thus, as illustrated in Figure 6.5 the lower bound for set of six elements will be LB = 5, that is, the total number of connections, except between the ends. Conversely, in the same figure, the upper bound for the same six-element set will be UB = 15. Hence, expressed mathematically the following two equations are produced.

$$LB = n - 1$$

Equation 6.8 – Lower Bound for the total number interactions in an assembly

$$UB = \frac{n(n-1)}{2}$$

Equation 6.9 – Upper Bound for the total number interactions in an assembly

Both equations can be easily worked out as the sum of an arithmetic progression.

6.1.2.3 Aiming for the boundaries

Plotting the equations just presented in the previous section results in a region surrounded by curves, as shown in Figure 6.6. This region is meant to be the zone that contains all possible combinations of interactions in any particular set of parts. For instance, a product with *n* components presents a total number of interactions LT. Considering the lower bound LB as the simplest possible arrangement, the level of 'saturation' can be deducted as follows.

$$LS = \frac{LT - LB}{UB - LB} * 100$$

Equation 6.10 – Level of interaction saturation

Saturation is referred to as the total number of interactions actually used by the product, from the entire range of possibilities, as suggested by the *Upper Bound Level*. In theory[24] a product could achieve its 100% level of saturation when all components are in contact. Designers could map the values for that particular product. This would help in the evaluation of the product complexity by allowing the comparison of two or more different configurations of the same product (e.g. diverse product structures). Conversely, designers could also compare products with similar numbers of components. In addition to this sort of comparisons, designers, with the use of this chart, could also determine the starting point for a series of product structure improvements.

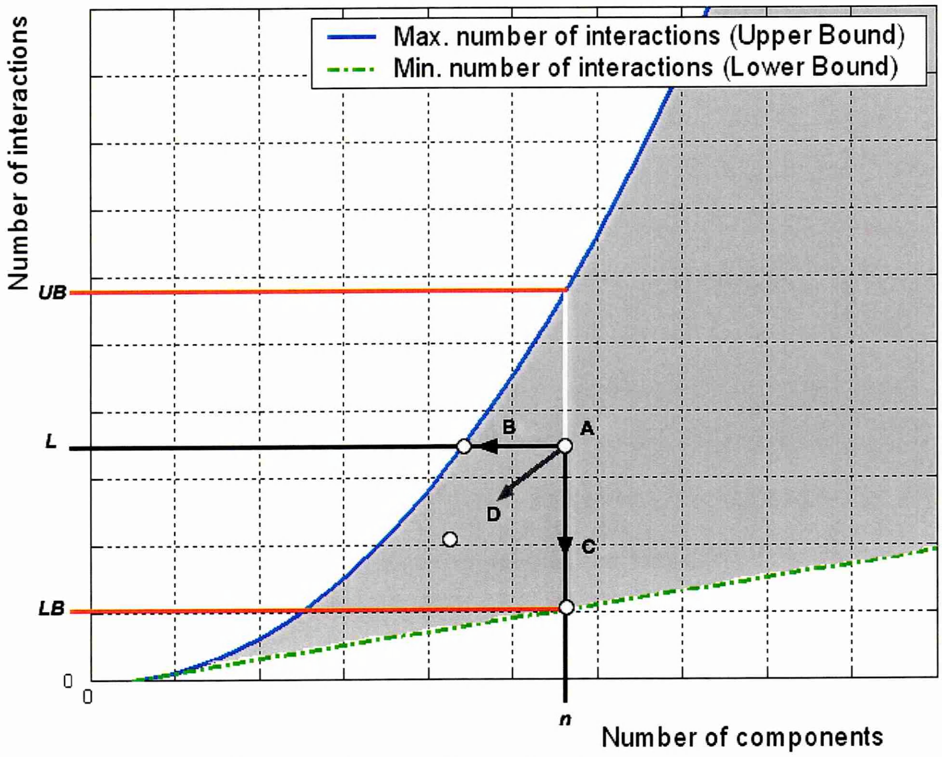


Figure 6.6 – Upper and Lower Bound for the number of interactions

There are four choices to start optimising the product structure as shown in Figure 6.6. The first one will be the redefinition of the product, yet keeping the same number of components and interactions. Second, reduce the number of components,

[24] The actual value for the upper bound level is always affected by other conditions, see section 6.1.2.4

without modifying the amount of interactions a great deal. Third, reduce the number of interactions by keeping the same number of components or reduce both interactions and number of components. Every option has various implications, showing the need for more additional metrics of complexity mentioned later on.

A. *Maintain constant the ratio of interactions to number of parts.* (Option A) – Trivial as it might sound, this option implies redefining the product structure in order to accommodate new parts, by re-designing existent elements yet keeping the amount of interactions constant. One reason can be that of adding extra functionalities to the product, usually introducing new components as existing components are re-designed.

B. *Try to achieve maximum level of connectivity. (Upper Bound)* (Option B) – Integrating components will certainly reduce the number of interactions. It can also occur that the number of components is reduced and the amount of interactions remains constant, actually producing the maximum number of interactions this –new product structure– can handle. This means that components are combined, hence increasing the ratio of interactions to amount of components as well. The geometric shapes of the newly created components will ultimately be affected, to accommodate their newly assigned tasks – integration equals higher interaction. This situation presents the designer with a new challenge, should he/she create a component from scratch with the burden of unproven shapes? Or should he/she re-design the existing component altogether, yet trying to lower the geometric complexity. This puzzle has an extra pitfall: The number of interactions is the same as the previous product structure; hence complexity in that direction has not been lowered.

C. *Try to achieve minimum level of connectivity (Lower Bound)* (Option C) – Keep the same number of components, re-design them in such a manner that they can have fewer interactions. This can be down to two options: (a) Components are re-designed such that they connect in common points (e.g. a "spherical" arrangement whereby all components interactions are the radii and share a common component for interaction, namely the centre). (b) Components are re-designed, but new components are added in keeping the number of elements constant and consequently, performing extra tasks.

D. *Try to reach the origin (0, 0)* (Option D) – Modify the number of components and number of interactions, at the same time. Ultimately, this option is the most commonly use. Nevertheless, this option requires a re-definition of the product structure. Components are not even combined or re-designed, they are completely changed. Yes, it does reduce the number of interactions and the amount of components, but then the design dilemma has been shifted. The project of building the new product turns into a whole new plan (i.e. new product, new approach – leaving no possibility of optimisation). However, at least it does encourage the designer to continuously test every design structure against this level of connectivity, instead of waiting till the end of the design process.

As mentioned in the introduction to this chapter, the trade-off between part-count and component shape complexity is an issue not addressed by common DFA practice. If the designer wants to reduce the number of components and interactions, option D is the best choice. But it requires a dynamic evaluation of the product structures, but to keep an optimum balance its value must be checked against other complexity metrics as mentioned later on.

6.1.2.4 Upper bound: Theoretical and Real

The theory previously presented needs to be adjusted. It needs to account for a more realistic approach to the maximum possible number of interactions. As the number of components increases, the possibilities of reaching the upper bound become dimmer. For physical reasons alone, the elements in any assembly are placed further apart reducing the likelihood of connecting with each other as show in Figure 6.7. This behaviour, for the real upper bound, becomes less predictable, its dependency on the product dimensions makes the number of components alone less relevant, but by no means negligible.

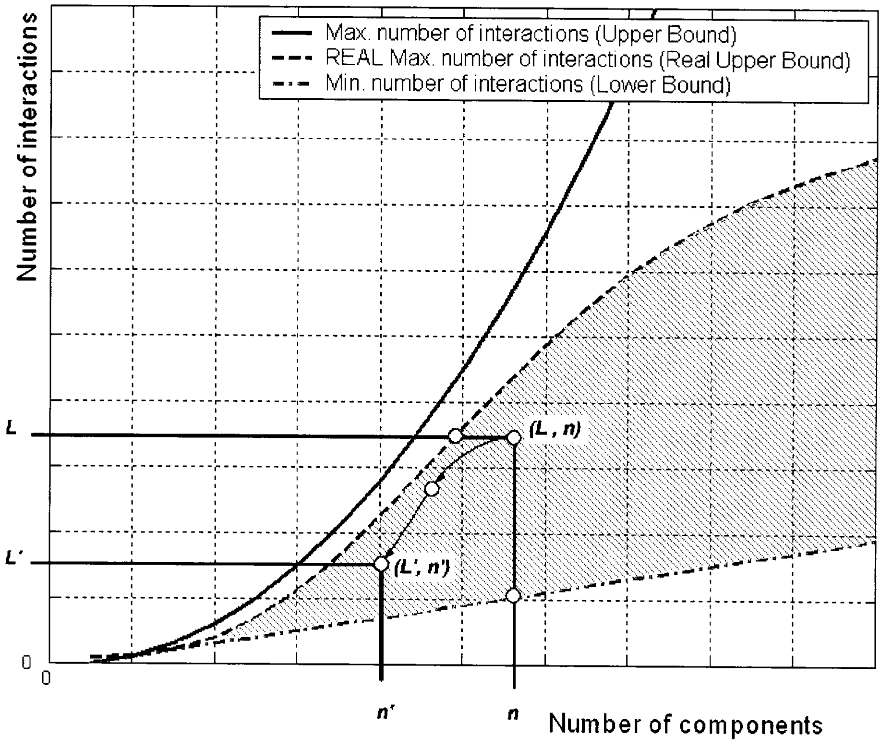


Figure 6.7 – Real Upper and Lower Bound - Modifications to the assembly

6.1.2.5 Variables needed to track interactions

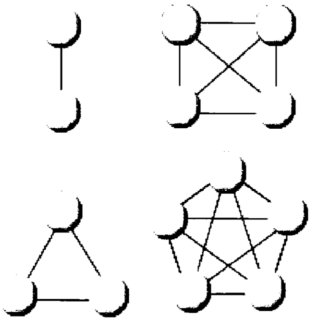


Figure 6.8 – Graph for the number of variables held at one time

Table 6.3 – Number of variables held at one time for a product due to pair-wise interactions alone

Components	*Total possible Interactions*[25]	*Overall*	*Overall + Arrangement*
2	1	3	4
3	3	6	7
4	6	10	11
5	10	15	16
n	n(n-1)/2	n(n+1)/2	1+n(n+1)/2

In the process of assessing this value for complexity, Table 6.3 represents the worst case scenario for the amount of variables that should be held at one time. From Figure 6.8, the amount of information that should be dealt with in real time can be represented by the following equation.

$$NVbles = 1 + n\frac{(n+1)}{2}$$

Equation 6.11 – Number of variables held at one time (worst case scenario)

[25] This is the Upper Bound Value

Obviously, Equation 6.11 represents the worst case scenario (a level of connectivity of 100%). A proper level or number of variables must be drawn up for every product. Nevertheless, the number of variables will indicate the maximum number of variables that must be held at least twice. Firstly, the variables are stored for the initial product configuration and, consequently, the variables that must be stored for the current or new configuration. In other words, as the design process evolves, several 'Product Structures (PS)' are created. The original or initial PS1 is analysed and stored, then after some modifications PS2 is obtained, analysed and stored. The process carries on until the final and more satisfactory PS is produced, which is then analysed and stored as well. During this procedure, at least two PS's must be recorded at one time in order to make their comparison possible. For a product with a small number of parts, this might not make any difference in the comparison processing time (or storage requirements). However, for a product with a large number of parts this certainly might be an issue. This is a situation that must be taken into account for an implementation in a software package.

6.1.3 Component Interfaces in an Assembly (Connection types)

It has been seen that the number of component binary interactions do give a measure of the complexity of the product structure, hence giving some support to designers. However, the exploration of types of interactions –herein referred to as 'interfaces'– further refines the complexity estimation. Mechanical interfacing can be classified into the following groups:

1. *System architecture and spatial adjacency* – Intuitively this classification relates the components' own geometry to their overall topology, as they are connected to each other in an assembly.

2. *Energy transfer* – Thermodynamics, Fluid mechanics and forces involved play a vital role in this type of interfacing. Energy of any type may be exchanged between components. Elements in a product are created and placed in the assembly to perform an exclusive task.

3. *Data and/or information exchange* – Along with the previous classification, this is a task based type of interface and, as its title implies, connects the elements according to the data they exchange.

Only mechanical interfacing is considered here, as mechanical[26] interactions are bound to influence the product assembly more severely than any other (e.g. digital, heat transfer, and so on). Following this rationale, component interactions can be then sub-classified into two different types: *Geometry and Adjacency Type (Contact),* a further classification such as *"Indirect interaction"* was originally planned, but as regarded as rather misleading and error-prone. Nevertheless, some information about kinematic analysis will be explained before introducing the two classifications.

6.1.3.1 Kinematic interactions – geometric information

Commonly, component binary interaction has been classified according to how it transmits motion –energy transfer– from one to the other. This classification denotes every component or machine part as a "link". It is generally used to carry a dynamic analysis, where the component that transmits the motion acts as 'driver' and the 'follower', unsurprisingly, acts as the output link or the link that receives the motion. These connections or joints are called *kinematic pairs* (or just pairs), because each pair consists of a pair of mating surfaces.

In this section, kinematic pairs are presented in a simple manner. As previously mentioned, this subject has been thoroughly studied in the past and numerous books have been dedicated to this topic[27].

Kinematic pairs are further sub-divided into '*higher*' and '*lower pairs*' based on the type of contact between any two components. In short, lower pairs (see Figure 6.9 and Table 6.4) are those such as the pin joint, where there is a contact surface between the pair elements.

[26] The term "mechanical" refers to structural or spatial types of interaction. Energy flow is another type of mechanical interaction, but it is often associated with heat flow, fluid flow, etc.

[27] For further reading: Uicker Jr., J.J., Pennock, G.R., and Shigley, J.E. (2003). "Theory of Machines and Mechanisms." 3rd ed. New York: Oxford University Press, Inc.; ISBN: 019515598x

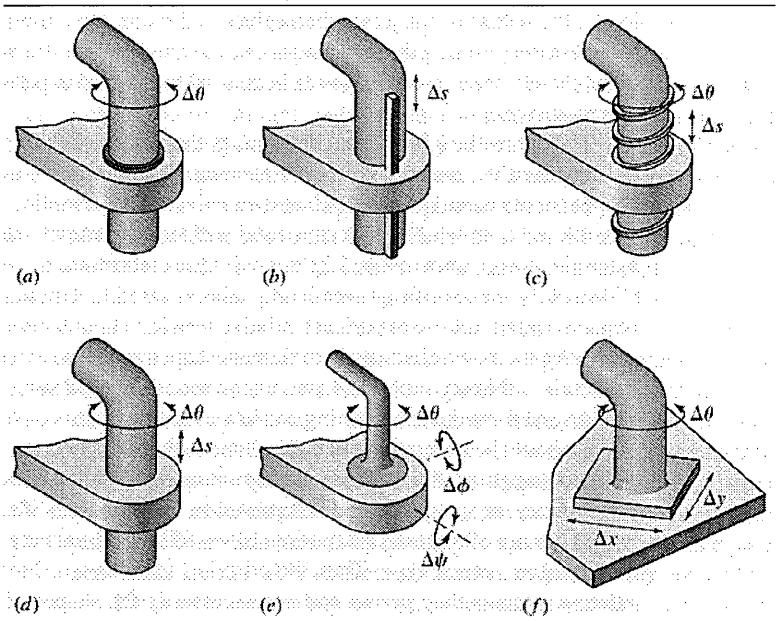


Figure 6.9 – Six Lower Pairs (a) revolute or pin; (b) prism; (c) helical; (d) cylindric; (e) spheric; and (f) planar. *(Adapted from Uicker Jr, Pennock and Shigley* ***[160]****)*

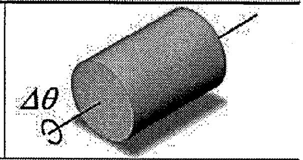
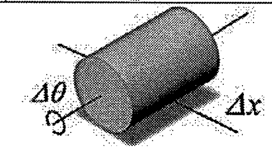
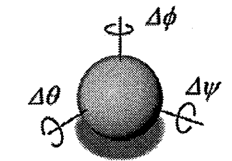
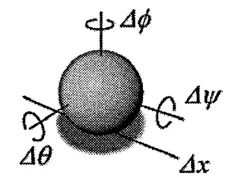
Table 6.4 – Lower Pairs (From Uicker Jr. Pennock and Shigley)

Pair	Symbol	Pair variable	Degrees of Freedom	Relative motion
Revolute	*R*	$\Delta\theta$	1	Circular
Prism	*P*	Δs	1	Rectilinear
Screw	*S*	$\Delta\theta$ or Δs	1	Helical
Cylinder	*C*	$\Delta\theta$ and Δs	2	Cylindrical
Sphere	*G*	$\Delta\theta, \Delta\varphi, \Delta\psi$	3	Spherical
Flat	*F*	$\Delta x, \Delta y, \Delta\theta$	3	Planar

Whereas higher pairs (see Table 6.5), such as the connection between a cam and its follower, have a *line or point of contact* between the elemental surfaces. However the classification system was further revised (See Ulicker *et al.* [160]) and modifications were introduced, such that the type of contact was not the only criteria to be used, but also the type of motion.

This information about kinematic pairs will be used in the following sections to help classify the interactions between components. This classification is based not only on the type of motion, but on the geometric information available.

Table 6.5 – Higher Pairs Joints

Description	*Pair variable*	*Degrees of freedom*	*Typical form*
Line contact between cylinder and plane (non-sliding)	$\Delta\theta$	1	
Line contact between cylinder and plane (sliding)	$\Delta\theta$ *and* Δx	2	
Point contact between ball and plane (non- sliding)	$\Delta\theta$, $\Delta\phi$, $\Delta\psi$	3	
Point contact between ball and plane (sliding)	$\Delta\theta$, $\Delta\phi$, $\Delta\psi$ *and* Δx, Δy	5	

6.1.3.2 Geometric interactions

Interactions amongst components might be grouped in a non-exclusive fashion, that is to say, one component can interact with another in two distinct manners, making it a good example of more entwined relations.

Geometric interactions are mainly related to the way the parts face each other, therefore, by naming them as links and/or interfaces two different characterisations can be made:

a. **Geometric Link – Docking:** Geometric interactions, in which parts play a "male-female" role but are self contained in their joint, in other words, no extra components, are required to tie them together. Figure 6.10 shows two representations of this type of interaction.

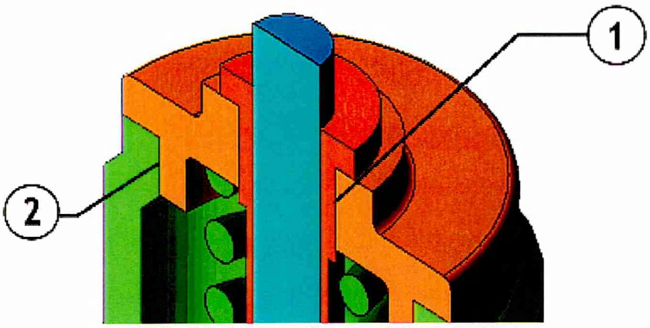


Figure 6.10 – Geometric docking

b. **Geometric Chain – Coupling:** Geometric interactions, in which parts play a "male-female" role, but require a third component to keep them together, as shown in Figure 6.11, where (1) shows the geometric interaction and (2) shows the third component to keep them together. Differentiation should be made between "geometric coupling" and "parts in contact." Geometric interactions as such should be considered as static or fixed, that is to say, this type of interactions should remain as such throughout the whole life to the product.

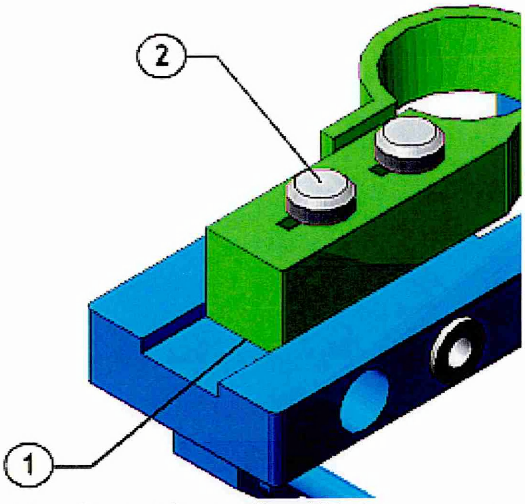


Figure 6.11 – Geometric coupling (third component required)

c. **Geometric dimensioning and tolerancing – Interfaces:** Dimensional tolerance is yet another level of information that can be extracted from the product. It can also be added to the reasoning for a valid complexity index. Even at an early stage, tolerances can and should be considered, either for process planning purposes or just for design requirements. Some design guidelines suggest that unnecessarily tight tolerances that are beyond the capability of the manufacturing processes should be excluded. Nevertheless, tolerance stack-ups in mating parts, whether the tolerances are tight or loose, add an extra layer of complexity to the design. This extra variable ought to be considered, not only for the manufacturing process to select from, but for the overall assembly.

6.1.3.3 Adjacent parts (Contact)

Not all parts have a geometric match, and not all parts will be fixedly connected to one another throughout the life of the product. There are kinematical interactions, constraining relations and even sporadic connections that need to be taken into account.

Grouping these interactions does not necessarily require complete geometric data and can therefore be indicated during the definition of the product structure.

a. **Tribocontact – Kinematic pairs:** A tribocontact type of interface is for parts with surfaces interacting in relative movement, that is, mechanical contact that involves friction and wear; as a result lubrication might be an issue.

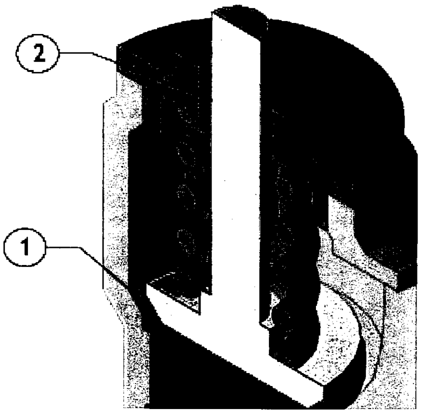


Figure 6.12 – Tribocontact (Friction & wear)

Figure 6.12 shows tribocontact interfaces two distinct variations: (1) Stem and seat in contact, but their mode of operation (open/close) requires that they must be able to have relative motion – separation. (2) Stem and bush in permanent contact, however due to the displacement needed for the stem to work, it involves friction and therefore there are tribological aspects to bear in mind.

This form of interfacing could be classified as an intermediate type between 'geometric interaction' and purely 'in contact', because most kinematical relations bear some geometric resemblance. Nevertheless, due to their relative motion, parts are dealt with in and associated to different levels of complexity. In terms of mechanism analysis, these tribocontacts can be considered as 'kinematic pairs'[28], and consequently be categorised as lower pairs, if two elements are in surface contact, and higher pairs, if the contact is at a point or along a line. Joint (2) in Figure 6.12 is a clear example of lower pairs; where as higher pairs are typified by gears and cams (see section 6.1.3.1 for kinematic pairs)

b. **Constraint:** There are no geometric restrictions or relations, other than coincident planes (lines or points). As shown in Figure 6.13, components A and B constrain each other from moving, their only geometric similarities are those of having planes in contact, but their overall geometric shapes are independent of each other.

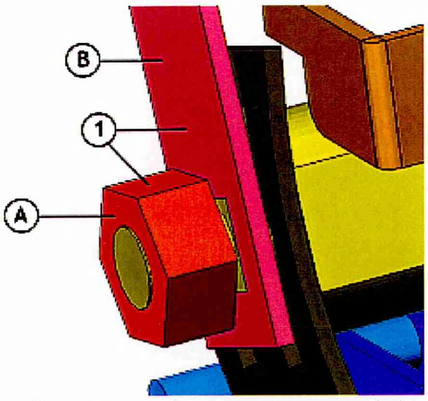


Figure 6.13 – Components in contact constraining each other

c. **Coincidental or sporadic (Locking):** Components interact coincidentally when they happen to be next to each other in an unintentional manner. It can be considered trivial, but this type of behaviour can signal either a bad design from the start or that components are intended to interact occasionally, see Figure 6.14. Such connection starts being of great relevance when parts wear down and collide producing extra wearing and friction. Parts that interact occasionally add the extra burden of synchronising components. Perhaps that is the reason why components get added to the product structure, to support other components, that one task alone being their only reason for existence. It is clear then, that the more components are added, the more interactions will appear, hence higher overall complexity.

[28] As mentioned before, in mechanisms and robot configurations, kinematic pairs are defined as the joints connecting two elements.

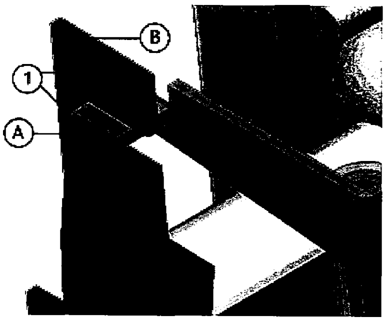


Figure 6.14 – Elements in sporadic contact

6.1.3.4 Indirect interaction

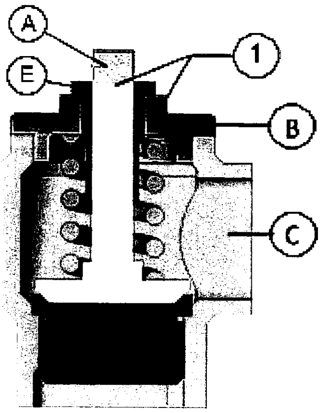


Figure 6.15 – Indirect component interaction

The reader could be excused for wondering whether a third classification of part-to-part interaction should be contemplated. This would be the case of an "indirect interaction" between components (*transitivity of interactions*). To clarify this point, examine Figure 6.15. In this illustration it can be seen that the internal diameter of the 'cover B' depends on the diameter of the shaft of 'stem A'. However, these two components are not in contact, therefore they do not follow either the geometric or the adjacent type of interaction. How can this sort of modifications be tracked down?

Perhaps this is the hardest type of interaction to predict, and it usually requires functional assessment to be able to establish it. It could be worked out by following geometric relationships amongst the components involved (Stem A, Cover B and Bush D in between) and therefore an indirect estimation would be unnecessary, because it would be contained in the traced interactions (A to D and D to B). It can be said that the whole assembly is bursting with indirect interactions, for instance, component A has an indirect impact on the shape of component B and perhaps on that of component C. Ill-defined component-to-component interfaces do not allow this sort of modifications to be accounted for. In a traditional component-oriented environment components are defined and then assemble, modifications to one component disrupt the assembly, but they are not transmitted down to the component level. In an assembly-oriented environment interfaces are defined and then the shape of the component is formalised based on its intended functionality and the set of interfaces contained within component. In an assembly-oriented environment modifications made to component A would be either "absorbed" by the shape of component D or transmitted to component B, but accounted for at all times.

Levels of indirect interaction: If it were necessary to estimate the indirect interaction between any two components, then to estimate the levels of indirect interaction, one would have to consider every pair of components (see Figure 6.15, components A and B) that are in direct connection with a third in between (component D). This could be said, is the first level of indirect interaction. Subsequently, 'B' is in direct connection with another component 'C', if B has been modified, then it would probably affect the shape of C, and therefore it constitutes a second level of interaction. The reader can see that it will soon be a problem to keep track of the different levels. This situation can be avoided by simply representing the types of interactions with the types defined above (Geometric and Adjacent)..

6.1.3.5 Indirect interaction made irrelevant - Discussion

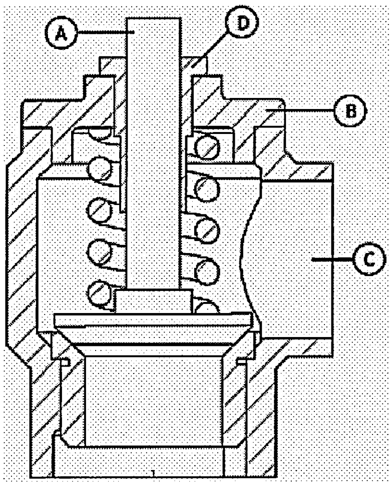


Figure 6.16 – Indirect component interaction vs. Geometric interaction

From the previous section, it can be said that indirect interaction is nothing but a redundant classification of "geometric interaction", and it is. Consider two different cases from Figure 6.16.

In the first case, components A, D and B can be all related through a "geometric linking or docking" interaction. That is, A and D share a docking type interaction, same between B and D, therefore the interaction of A and B has been established by this chain of connections, rendering their "indirect interaction" classification as pointless, i.e. any changes to the shape of component A, is transferred to component D, which in turn suggests a modification to the shape of component B. Put in this way, the 'indirect interaction' classification has been made redundant. All interactions can be followed up by a "chain of interactions," Now, consider a second and different situation, for components D, B and C. It is still valid that elements D and B share a 'docking' type of interaction, as it is between B and C. However, changes to the shape of D do not necessarily affect that of element C. This modification to the shape of D is transferred to B, through their mutual interface, but the shape of B could "absorb" such modification, whilst keeping the interface B-C intact, thus the shape of C is unharmed.

Consequently, the correct estimation of an 'indirect interaction' cannot be resolved for every situation. This ambiguity inevitably introduces a level of uncertainty and subjectivity incompatible with the proposed set of metrics, which need to be applicable and objective throughout the design process.

Nevertheless, one way to go around this ambiguity is through a functional assessment of components. This assessment will somehow reflect the effects of modifying the shapes of the components, thus highlighting whether changes made to one component will propagate through the entire product structure. Since there is not a clear way of stating a criterion that objectively differentiates an 'indirect interaction' relation from the two already presented (i.e. geometric and adjacency), this 'indirect interaction' classification will be avoided.

6.1.3.6 Non-exclusive component interactions

As mentioned above, components can interact in such a way that their relation can be classified as different types of interactions at the same time. Figure 6.16 shows components A and D interacting as "Geometric Link or Docking", but their relative movement also places them in the category of "Tribocontact".

It is worth mentioning that not all 'tribocontact' interactions present a 'geometric docking', a basic example might be that of a roller over a plane. Their relation is purely frictional, but there is no 'male-female' matching amongst the faces involved (once again, refer to section 6.1.3.1 for information about kinematic pairs)

6.1.4 Complexity of individual interfaces

Component interfaces have been categorised to be able to explore their impact on the interaction between any two components. A ranking procedure puts numerical values to this effect. It has been selected according to the level of detail required for every interface, there might not be a unique way to weight every interface, but as long as the same scale is used, it should not present a problem.

Table 6.6 and Table 6.7 present a set of values selected for the different type of interfaces, based on their geometric data and their adjacency condition, respectively. Information such as kinematic pairs and geometric data were used as the foundation of this scale. It still reflects a subjective weighting system, but it is meant to be extended or improved in subsequent studies of component interfaces.

In Table 6.6, the values have a penalty, according to their subdivision, for instance, geometric docking has a penalty equal to 1.0, whereas geometric coupling, because it involves extra components has a higher penalty value. This penalty value is further enhanced in its accuracy by using the actual geometric data within the interface.

A CAD model is generally based on geometric and topological information (see Chapter 3: section on shape complexity). Geometric data can be extracted based on the number and types of faces (e.g. planar, cylindrical, conical, toroidal, to name a few). If there is no geometric data, the penalty values would suffice for the time being.

Table 6.6 – Nomenclature and numerical values for component interfaces

Interface Type		Identification	Penalty	Interface Shape (§)	Value 'Hg'
General	*Subdivision*				
Geometric	Docking	G1	1.0	Geom. Data	= Penalty * Interface Shape
	Coupling	G2	2.0	Geom. Data	= Penalty * Interface Shape

§ → Interface shape – Value associated with the geometric data available for this interface. It can be produced tentatively, based on the intricacy of the geometry, or it can be produced more accurately once the manufacturing information of the entire component is known[29].

On the other hand, Table 6.7 is based on the kinematic information of the connections. The reader can be forgiven for thinking that it is hard to compare between two types of interfaces, that is, from the table one can read that a tribocontact interface of the type 'helical' is as complicated as a constraint interface

[29] Manufacturing information is extracted once the component has been thoroughly defined. Interface definition is just one of the steps, the other being its intended functionality.

of the type 'intentional'. Once again, these values reflect a subjective approach, which can be further refined. However, the classification of joints based on the number of degrees of freedom has been exploited as a ranking value, which is why helical, circular and rectilinear interfaces have a value of '1.0', this value represents the one single degree of freedom available in such connections.

Table 6.7 - Contact/ Adjacency estimations for component interfaces.

Interface Type			Identification	Value 'Ha'
General	*Subdivision 01*	*Subdivision 02*		
Contact (or Adjacency)	Tribocontact	Circular	Lower_A1.01	2.0
		Rectilinear	Lower_A1.02	2.0
		Helical	Lower_A1.03	2.0
		Cylindrical	Lower_A1.04	3.0
		Spherical	Lower_A1.05	4.0
		Planar	Lower_A1.06	4.0
		Cylinder-plane non-S	Higher_A1.07	2.0 * 1.5
		Cylinder-plane S	Higher_A1.08	3.0 * 1.5
		Ball-plane non-S	Higher_A1.09	4.0 * 1.5
		Ball plane S	Higher_A1.10	6.0 * 1.5
	Constraint	Intentional	A2.01	2.0
		Unintentional	A2.02	2.5
	Sporadic		A3.01	3.0

The values or 'weights' assigned to the contact/adjacency type of interactions is higher than those initially given to the geometric interfaces. The reason for this higher value is because geometric interfaces leave a "reminder" in the component's shape. This geometric information continuously reminds the component that it belongs to a greater structure (assembly). An adjacency-type does not leave this "memento", therefore with this type of interface components are not marked with a reminder of the existence of neighbouring components.

In view of that, Table 6.6 and Table 6.7 give the values of 'Hg' and 'Ha' that will be used in Equation 6.12 (value of H_i) (section 6.1.5.1.). Furthermore, this scale will help to report on additional information to the user. Components with intricate interfaces and larger number of them can be singled out. Designers can be given advice as to where to lead his/her design optimisation methods. It can show that a component might have a large number of interactions, but its interfaces are fairly 'simple' (that is, not very strong), thus not requiring the dedication of resources required in another component with fewer interactions, but of stronger type.

6.1.5 Matrix Arrangement of Interactions and Interfaces

Two of the most commonly used forms to represent product architecture, and assembly related information, are graphs and matrices. A graph-based[30] representation (Figure 6.17) is ideal for illustration purposes and for information visualisation – product manipulation (in terms of creation, modification and handling) is also convenient as every component has a symbol and every relationship can be 'seen' straightaway.

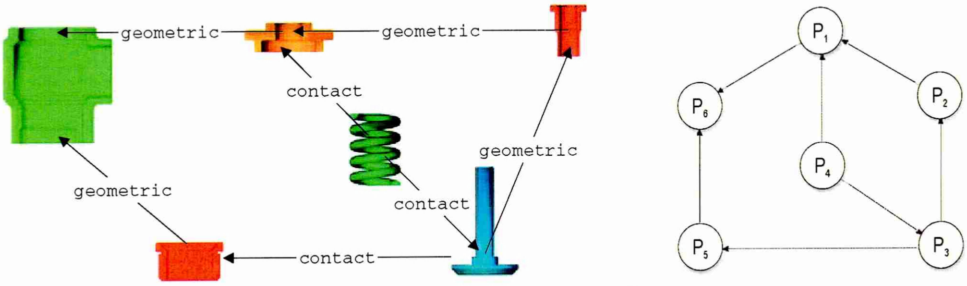


Figure 6.17 – Graph representation of interactions and interfaces

On the other hand, matrix-based representations can be manipulated more easily, either within any commercially available spreadsheet package or a few lines of code with arrays and loops[31]. Matrix-based representations allow the extraction of information about a single component by looking up its row or column. The presence (or absence) of relationships between any two components can be inserted in the matrix, eventually forming a symmetric matrix –components are equally placed in rows as in columns. (see Appendix A).

There is no need to have any assembly sequence at this stage. However, the product structure must be known, that is, with a well-defined functionality that implies a certain number of components to perform the intended tasks. Nevertheless, the product structure might not be entirely determined, in other words, components do not necessarily have to have a particular form yet. This allows the insertion of 'concepts' or 'components at conceptual stage' without geometric data attached. The product architecture thus selected will be used as the basis for extraction of complexity metrics.

[30] It is not relevant whether, as in this case, the graph is directed or undirected, providing that at this point, there is no particular interest in knowing the products particular organisation.

[31] Matrix and graph representation can be easily encoded in any programming language and there is abundant information about graph algorithms and matrix manipulation.

Provided that not all components will be thoroughly defined, the following layout will be used internally to represent all the interactions amongst the different components. Initially, the components are set up in rows and columns without any predefined order (see Figure 6.18). For the sake of illustration only the first and most visible types of interfaces will be portrayed, but a more detailed outline will be presented later on.

Name / Component ID		1	2	3	4	5	6	7	8	9	10	**a**	**b**	**c**	**d**	**e**	
Component 01	1	**1**	**b**										1				1
Component 02	2	**b**	**2**	**b**	**a**	**c**						1	2	1			4
Component 03	3		**b**	**3**	**b**	**e**							2			1	3
Component 04	4		**a**	**b**	**4**	**a**						2	1				3
Component 05	5		**c**	**e**	**a**	**5**	**A**	**A**	**d**			3		1	1	1	6
Component 06	6					**a**	**6**		**a**	**a**	**a**	4					4
Component 07	7					**a**		**7**	**a**	**a**		3					3
Component 08	8					**d**	**A**	**A**	**8**		**d**	2			2		4
Component 09	9						**A**	**A**		**9**	**d**	2			1		3
Component 10	10						**A**		**d**	**d**	**10**	1			2		3
Geometric Docking	**a**		1		2	3	4	3	2	2	1	**18**					
Geometric Coupling	**b**	1	2	2	1								**6**				
Adjacency: Tribocontact	**c**		1			1								**2**			
Adjacency: Constraint	**d**					1			2	1	2				**6**		
Adjacency: Sporadic	**e**			1		1										**2**	
Total No of Interactions		1	4	3	3	6	4	3	4	3	3						**34**

Figure 6.18 – Matrix representation of component interactions in binary space

Any two components P_i and P_j are observed and their types of interfaces annotated, following the *i-th* row and *j-th* column for this purpose and likewise for the *j-th* row and *i-th* column, as the matrix is clearly symmetric. A summary of the total number of any type of interfaces follows at the end of the matrix columns (and rows). The same can be said for the total amount of interactions per component.

However, Figure 6.18 presents a binary space of interactions, where no other information but type of interfaces and the number of them is revealed. These data are, nonetheless, very useful for the evaluation of the 'level of connectivity' amongst the components. This level of connectivity is the number of interactions per component in the whole assembly and how much it deviates from the minimum level of connectivity (see section 6.1.2.1 above).

An enhanced presentation of the 2D space portrayed gives the designer the possibility of weighing up the different interfaces presented as per component interaction, creating a multi-dimensional space where more variables are considered. This will be clearly presented in section 6.1.5.1.

Furthermore, this dual space/matrix is a mere internal representation of the data being manipulated, and thus hidden from the user. Information, instead, will be extracted from the product structure as the designer defines it. In any software implementation this extraction of information is expected to be an operation running in the background. (The way data are subsequently fed into the matrix is explained in Appendix A.)

6.1.5.1 Components interacting in a multi-dimensional space

In the previous section, a matrix-based representation was chosen, such that binary interactions were accounted for by extracting the information regarding the interface type. These data, however, just included the type of interface, whether geometric or adjacency, but no additional data were stored. Then again, this information can be enhanced with the introduction of weights. Interface weights show how difficult in achievement a particular interface is, and therefore requiring more resources for the assembly and manipulation of the components.

With the aim of getting a more realistic measure of the interface in question, suppose the following scenario. Imagine a space as the one illustrated in Figure 6.19, in which the grid (floor) represents the matrix arrangement of interactions – only half of the grid is shown as the matrix is symmetrical.

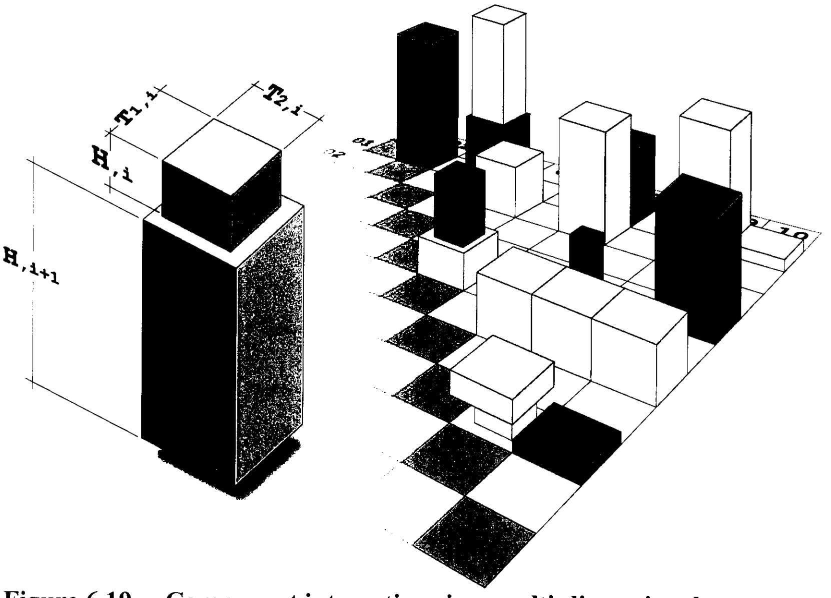


Figure 6.19 – Component interactions in a multi-dimensional space

Every interaction will be symbolized by a box highlighted with a different colour for every different type of interface. The boxes can be stacked up as more and more different types of interfaces are found in the same interaction. The dimensions of every box have a special meaning. The height corresponds to the numerical value assigned to that particular interface[32]; the cross section area denotes the level of refinement given by the level of the tolerances for the manufacturing processes involved in the making of that particular interface. Ideally, the cross section will be square, i.e. both tolerances made equal, but in case the components have different tolerances, for that individual interface, then the length of those sides must be considered separately.

The weighted value for any particular interface will then be denoted by the volume of every box times the type of lubrication required, so as to account for the weight of the box itself and the assessed value of the whole interaction, by the sum of all weights of the boxes compromised. Therefore, as pointed up in Figure 6.19, for any interaction *l*, there might be a set of interfaces I = {v_1, v_2 … v_i}, where the value for every weight v_i will be expressed by:

[32] Refer to section 6.1.4 for the procedure to establish numerical values for every type of interface.

$$v_i = \rho_i \cdot H_i \cdot T_{1,i} \cdot T_{2,i}$$

Equation 6.12 – Weight of every type of interface compromised in any component interaction

Where:

v – Assigned weight to every interface,
ρ – Type of lubrication,
H – Numerical value estimated for the interface considered (either Hg or Ha) (*These values can be read from Table 6.6 – Nomenclature and numerical values for component interfaces*),
T_1 – Level of tolerance for the interface in component 1, (inverse of)
T_2 – Level of tolerance for the interface in component 2. (inverse of)

As mentioned above, the values T_1 and T_2 can be made the same given the manufacturing tolerances in the interface between the two components is identical. But as for many cases the manufacturing tolerances can be dissimilar if a component does not discriminate between the individual interfaces and the overall component manufacturing tolerance.

Equation 6.12 shows that for tight tolerances, the cross sectional area is large, where as for loose tolerances the area tends to narrow down – the tolerance level is inversely proportional to the tolerance value. On the other hand, the value assigned to rho (ρ) deals with the tribological factors involved. In machine design, some joints need to be lubricated and kept enclosed, for this kind of contact lubrication with grease is the preferred option. In some other cases, lubrication with oil is the appropriate option, either for ease of access or constant lubrication (e.g. cooling).

Consequently, the weighed up value for every interaction, will be the sum of every weight of all the interfaces involved, as conveyed by the following equation:

$$w = \sum_i v_i = \sum_i (\rho_i \cdot H_i \cdot T_{1,i} \cdot T_{2,i})$$

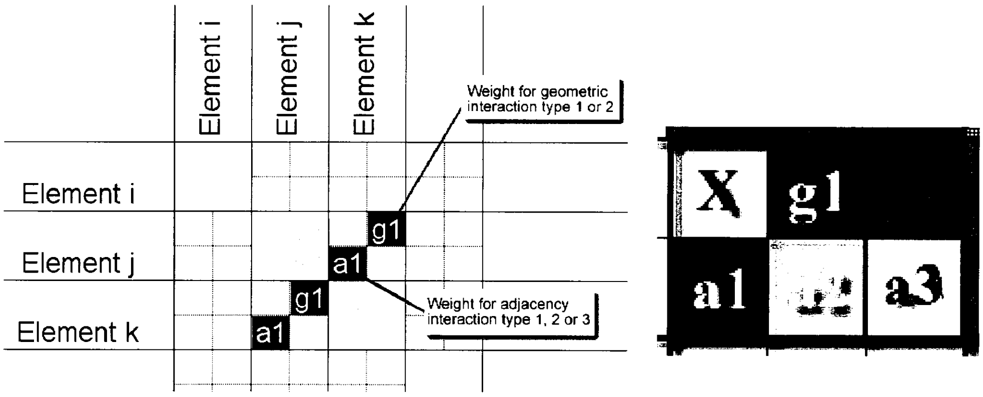
Equation 6.13 – Weighted value of every interaction, for all the interfaces involved

In cases where interfaces are not well defined or require no lubrication, default values for T and ρ must be used (hence by default T = 1.0 and ρ = 1.0).

Values assigned to the tolerance level are highly dependent on the type of material used for the component, and consequently on the manufacturing process selected. Surface finish conditions the default tolerance value. Information about the manufacturing process further refines this information, improving the allocated tolerance value.

Example: Suppose that two components are to be adjacent to each other. It is expected that the material to be chosen, for both components, will be 'stainless steel'. The manufacturing process is moulding, which can reach tolerances of 0.0127 mm (or 0.0005 in) or less. The tolerance level is set to be inversely proportional to the tolerance value, therefore T = 1/0.0127 (or T = 78.74).

6.1.5.2 Amount of information in a matrix-based representation


Element i
Element j
Element k
Weight for geometric interaction type 1 or 2
Element i
Element j
Element k
g1
a1
g1
a1
Weight for adjacency interaction type 1, 2 or 3
X
g1
a1
a3

(a) (b)

Figure 6.20 – Information contained in a cell; (a) Component to component; (b) Bits of information per cell

The amount of information that can be allocated at a particular cell is defined by the number of interfaces involved. However, since only one of the geometric types, plus one of the adjacency types can be selected at one time, then it carries "2 bits" of information from that selection, at most (Figure 6.20).

However, since other data can be included (i.e. tolerance and geometric shape index), the amount of information can increase presenting a more accurate picture of the complexity of the interface. This is due to the 'multi-dimensionality' of the interface assessment studied in the previous section.

6.1.5.3 Component clusters

One benefit of having a matrix-based representation of the product structure is the convenience to group components into clusters. In chapter 4, the design of modular products explored this component clustering capability (DSM methodology – Eppinger *et al.* [127])

In short, component clustering consists in re-shuffling the matrix-based representation, so that components that are related stay together, this is immediately reflected in “clusters” of component relationships form along the diagonal. For any product structure, this procedure can be followed, sometimes suggesting the arrangement of sub-assemblies, which can be put together in parallel (see Figure 6.21 – shown for illustration of cluster formation only).

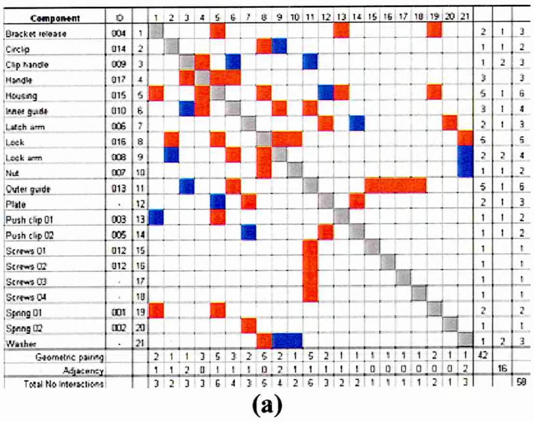


Figure 6.21 – (a) Original matrix representation of a product structure, (b) Re-shuffled matrix to detect component clusters. (*Taken from case studies in Appendix B*)

6.1.6 Periodicity – Subassemblies, arrays and pattern formation.

Products do not normally have all-unique components. Basic arrays of fittings, such as the universally discouraged bolt-washer-nut array, more often than not, populate product structures in such a way that they have a propensity to create patterns. The discussion of pattern formation, in this chapter, falls into the domain of ‘patterns of interaction in a product.’ [161]

Any measure of complexity which is proportional to the number of interactions may be sent “off-scale” because of the formation of patterns of interaction. Their appearance in the product structure will ultimately be reflected in an overstated level of complexity. This can be corrected by detecting the template that created such pattern, which in itself has a certain level of complexity due, primarily, to its own interactions. Once revealed, any other similar occurring instances will have the advantage of a known procedural style, thus reducing the complexity level. In other words, the repetition of specific configurations within the product structure immediately denotes a level of complexity lower than that of a product without recurrent, repeated or redundant component arrays.

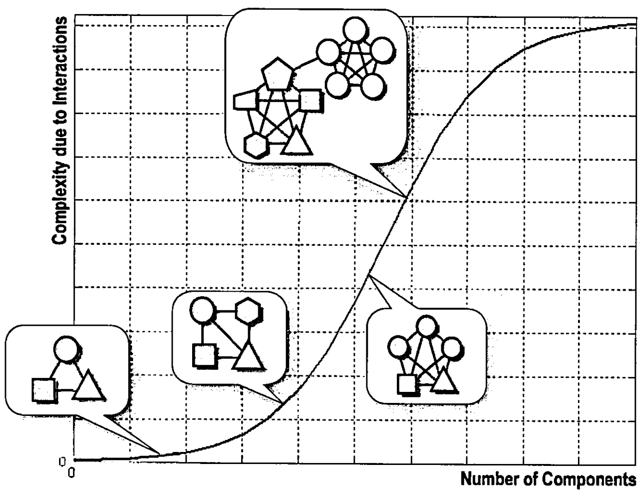


Figure 6.22 – Complexity affected by arrangement periodicity

Figure 6.22 presents the increase of complexity within the product as components are constantly inserted into the structure. It is expected that a "stagnation point" (change in curvature) in component 'diversity' will be reached, giving way to a behaviour of component monotony (plateau). Although this is only a hypothesis, this flattened behaviour will show that components will not present further dissimilarities, as they share more and more common features, to accomplish the same task.

6.1.6.1 Component types

Repetition or component recurrence makes the notion of 'component types' a relevant concept. Component types represent as an extra layer of complexity. Instantiation of components has been commonly used in CAD systems to model assemblies. Elements (component models) are created once and stored in a separate file. If one needs to be part of an assembly, then an instance of it is created and placed in the assembly. Every time a component (that already exists within the assembly) is required, the same procedure is repeated. Should the original component ever be modified, then its modifications would reflect in the assembly that has instances of it. This practice is a lot more economical than a "brutal copy" of the file (component) into the assembly, not only because it does not update the component after any modification (no link between assemblies and components), but because it increases the size of the file that contains the assembly.

If instantiation is a simpler approach, then it ought to be considered in fine-tuning the complexity metrics that depend on the component interactions alone, thus making them reflect a much more real level of complexity of the product.

The amount of dissimilar component types depends on the number of components within the product structure. The designer is the one responsible for establishing this amount –not intentionally, perhaps– but by the introduction of dissimilar components into the assembly.

Nevertheless, there must be a limit for the number of dissimilar components types. All components would start to share more and more common features, thus losing their differences. It is predicted that for any given product (or configuration), there will be a state at which the variation of components reaches a *plateau*. Figure 6.23 is a rough representation of this behaviour. Yet a mathematical representation is rather elusive. Even so, there is, after all, an algebraic representation that reflects the impact produced by 'component type variation' in an assembly.

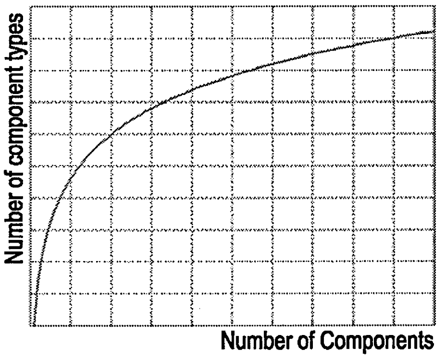


Figure 6.23 – Number of various types of components in an assembly

$$ACI_L = \frac{1}{2n}\sum_{i}^{n}\sum_{j}^{l} w_{ij}$$

Equation 6.4 – Assembly Complexity Index due to component interactions – Second approach (Recalled)

The relationship between the "index of assembly complexity" (presented in Equation 6.4 –recalled below) and "the number and nature of the components" can be adjusted with the introduction of a moderating "component variant factor" (Equation 6.14)

$$Vf = \frac{m}{n}$$

Equation 6.14 – Component-variant factor registers types of components found in an assembly

Where: Vf - Variation Factor,
M - Type of components (amount of),
N - Total amount of components

DFA methodologies encourage variant reduction by redesigning components, either by the combination (of two or more components) or by simply creating new parts altogether. This variant factor makes the ACI reflect a much more realistic measure of the intricacy in the product. It puts a damper, so to speak, on the Assembly Complexity Index. It is worth mentioning, nonetheless, that the amount of different components in an assembly depends ultimately on the decisions made by the designer.

During assembly, components are manipulated, inspected and handled such that they create a unique product structure. Component repetition or recurrence immediately establishes a handling practice that can be reused for every component. Consequently, the complexity of the product is lowered by the decrease in the amount of dissimilar component types.

$$ACI_{L,VF} = VF * ACI_L = \frac{m}{2n^2}\sum_i^n \sum_j^l w_{ij}$$

Equation 6.15 - Assembly Complexity Index due to component interactions and their variations

6.1.6.2 Standardisation

A logical consequence of taking into acoount the periodicity of components in assemblies is that it lowers product complexity. A characteristic of the product, considered constant for a given product structure, can be actually lowered by modifying the type of components, yet keeping the same number of them. Standardisation, as a means of reducing complexity and component variants, actually boosts the manufacturability of the product itself. It also increases the chances of automated assembly, for it presents a repeated mode of assembly.

6.1.6.3 Re-usable information

As expressed in chapter 3, structural and information-related complexities are two different ways of looking at the same thing. Periodicity, seen as a matter of information theory, can be regarded as a way to reduce information in the assembly process. The assembly process is composed of a set of instructions for putting components together. Instructions such as component position and orientation, next component, etc. are commonly found in sequence analysis and procedures. Information regarding neighbourhood, interfaces and interactions are commonly found in architectural decomposition of the assembly (product). Considering periodicity in assembly complexity makes it possible to distinguish between the amount of instructions needed for one component, also regarded as specific component information, and the amount of instructions needed for recurrent components. For a component used once in an assembly, instructions are created and used once, whereas for recurrent components, assembly instructions are produced and recycled, with the obvious benefits.

6.2 Case studies on connectivity and interaction

Since the core of the development of useful complexity metrics is the evaluation of component connectivity and interaction in the product structure, six case studies have been selected to show the benefits of applying product complexity metrics. It presents a heuristic approach, where the cases proposed do not represent a statistical, and nor a mathematical proof of the metrics, but rather a verification of the results and improvements suggested theoretically throughout this chapter.

The case studies are extracted from a set of products evaluated with the Lucas DFA methodology (except one). An original and suggested re-design is presented with the aim of evaluation the reduction (or increment) in product complexity and complicatedness after following DFA guidelines. This offers the added benefit of detecting whether the complexity of the assembly has been tackled and whether its complicatedness, if any, has been reduced. It also highlights the benefits that would bring a proactive implementation of such DFA guidelines.

The selected case studies are the following:

1. *Pressure relief valve* – Initially with a DFA design efficiency of 80%, this became a remarkable 100% in the redesign.
2. *Oil pump* – Originally with thirty-two (32) components, it was reduced to twelve (12),
3. *Staple remover* – The original design had eight (8) components and a DFA efficiency of only 12.5%. The suggested redesign has simply just one component. Visibly

4. *Motor drive* – It had a Boothroyd-Dewhurst DFA efficiency of 7.5% and 19 components. The redesign had seven (7) components and 26% of BD DFA efficiency,
5. *Wiper motor* – This motor was used for a rear screen wiper motor; it originally had thirty-one (31) components and was reduced to six (6). It also illustrates the by-product of the analysis with the adjacency matrix: identification of clusters for parallel assembly,
6. *Door latch* – Finally, a product with sixty-two (62) components (only twenty-one (21) are analysed) and an efficiency of 4.8% (Lucas method), it was improved to 22.5% and seventeen (17) components.

These studies and their results are presented in Chapter 7. It shows evaluations using the adjacency interaction matrix, graph representation of interfaces and a set of complexity metrics along with a list on comments on how they behaved according to their complexity and complicatedness.

6.3 Structural Complexity – dynamic

Finally, complexity metrics can be extracted from already existing studies, for instance assembly-sequence analysis [14, 162-167] and those dedicated to sequence complexity [168]. Assembly sequence planning or evaluation more often than not is carried out at the end of the product structure creation. A proactive DFA approach can benefit from the information already produced in the identification of component clusters mentioned in section 6.1.5.3.

6.3.1 Sequence complexity

It is well-known that assembly sequence and, therefore, sequence complexity are one of the most studied areas when dealing with assembly. It has produced a large amount of information, which can be used to extract valid metrics or means of detecting sources of complexity. This type of complexity mainly addresses the prediction of the most convenient assembly paths. Such predictions imply a certain number of possibilities that are directly influenced by decisions made at the design stage. As expected, they heavily depend on the architecture of the product, whereby adding new components, new possibilities of assembly are presented; this creates continuous bifurcations in the sequence analysis. Bifurcations are not only present in assembly sequence, for they actually populate the whole product – as already mentioned.

The previous chapter postponed the discussion of the group of sequence evaluation indexes presented by Barnes *et al.* [113], namely:

- *Insertion Index* – This is, as expected, based on the total number of insertion operations for each part, regardless of whether they occur in parallel.
- *Stability Index* – It considers the two different approaches to measuring sequence stability, geometric analysis and required fixing tools. The authors selected the stability index based on the account of jigs or fixtures necessary for each sub-assembly and one for the assembly of the overall product (no geometric data required).
- *Difficulty Index* – It is based on the calculation of component fitting indexes suggested by the Lucas DFA method, which combines an analysis of all aspects of the operational difficulty within an assembly sequence.
- *Complexity Index* – It evaluates the complexity of an assembly sequence based on the number of operations necessary to insert and secure one part to ready the partial assembly for the introduction of another component.

The author is aware that there are several other methods to determine the complexity of assembly sequence. The one presented above was chosen because of the variety of methods available. These can be introduced into the product complexity analysis framework. However, since assembly sequence analysis is a well-explored area, information is readily obtainable and therefore it will not be further discussed in this thesis.

Chapter 7

CASE STUDIES: ADJACENCY MATRIX EVALUATION FOR DIFFERENT PRODUCTS (RESULTS)

This chapter presents a summary of the analyses performed on several products that have been previously analysed using DFA methodologies (i.e. Lucas DFA and Boothroyd-Dewhurst methodologies). The results have been used to highlight the ideas proposed in this thesis. These findings also help to elaborate on the proposed theory of complexity analysis through component-interface discrimination. Every product has been analysed in a similar fashion, that is, it follows a regular procedure to extract information about the every product's configuration. This information is then evaluated using the complexity metrics suggested in this work and the results are commented upon. These studies present certain characteristics that help to stress the points made in chapter 6, namely: *vertical assemblability, zero vertical deviation, component variation discrimination*, amongst others.

Ideally, the suggested complexity evaluation should be performed at every stage of the produce development cycle. Once a concept has been formalised, a step-by-step design evolution can be monitored for factors affecting the product configuration. In this manner, it is possible to see whether the alteration of one factor has positive (or negative) effects on the product architecture -from the DFA standpoint. This analysis of "one variable at a time" can also be extended with multiple factor variations, that is, when two or more factors are modified (and annotated) and their combined effect visualised on the behaviour of the product architecture.

However, when only the original design (initial step) and suggested redesign (final step) are presented, it is difficult to see what improvements can be suggested. Moreover, it is not possible to establish what adjustment impacted the product architecture more dramatically. This is mainly due to the fact that suggested redesigns, as presented here, are often the result of a series of modifications. The outcome rarely bears any resemblance to the original designs. Consequently, the final product (redesign) is often seen as an entirely different product. In these circumstances, the complexity analysis does not help to indicate what particular modification turned 'Design A' into an improved 'Design B'. This group of modifications usually result in the loss of the link between 'Design A' and 'Design B'.

Regardless of these difficulties, the presented analysis manages to display the benefits of using a monitoring system. It is also possible to see that with very little information, much can be learnt about the future implications of the chosen configuration (once again from the DFA viewpoint).

Some case studies exhibit greater variations in one or two of factors (e.g. vertical assemblability, architectural compactness or Level of interaction saturation). These cases will be commented upon to especially draw attention to such disparities. Nevertheless all the cases present variations worth examining, as will be seen later on.

There are two specific issues that need to be addressed before presenting the analysis, and these are:

Interfaces estimation – Interfaces were classified only according to their major influence. Although, in chapter 6, it was stated that interfaces could be of the type 'geometric' and 'adjacent' at the same time, in this evaluation, only the leading type (either geometric or adjacent) was presented in the 'interface distribution graph.' In these examples, the actual geometric evaluation has been skipped (no actual interface weights have been assigned so far). Nevertheless, this proves that this metrics can be implemented even with reduced geometric information.

Assembly Complexity Index (ACI) [33] – As seen in chapter 6, this index is exclusively due to component interactions and their variations. However, for illustration purposes, the original influence of the interactions ("weights" of the interfaces) has not been computed, every interface has been considered to have the same influence, and its weights has been taken as unitary. This means that the original equation for ACI (see below), will be used with the value of every $w_{ij} = 1$:

$$ACI_L = \frac{1}{2n}\sum_i^n \sum_j^l w_{ij}$$

Equation 6.4 – Assembly Complexity Index due to component interactions – Second approach (Recalled)

$$ACI_{L,VF} = VF * ACI_L = \frac{m}{2n^2}\sum_i^n \sum_j^l w_{ij}$$

Equation 6.16 - Assembly Complexity Index due to component interactions and their variations (Recalled)

Under these circumstances (w = 1), the Assembly Complexity Index is transformed into the Unitary-weighted Assembly Complexity Index:

$$ACI_L(unitary) = \frac{1}{2n}\sum_i^n \sum_j^l (1) = \frac{L}{2n}$$

Equation 7.1 - Assembly Complexity Index, with unitary interface weight.

Notice that **Error! Reference source not found.** for ACI$_L$ is actually the same as the connectivity ratio (L/n) divided by two; hence the unitary-weighted Assembly Complexity Index that takes into account component variance will be represented as in Equation 7.2.

$$ACI_L(uni-weighted) = VF \cdot ACI_L(uni-weighted) = \frac{m}{n}\cdot\frac{L}{2n} = \frac{m \cdot L}{2n^2}$$

= [(Variant factor)*(Connectivity ratio)/2]

Equation 7.2 - Assembly complexity index, with unitary-weighted interfaces and Component variation factor.

[33] This equation has been explained and devised as shown in chapter 6. It is recalled for illustration purposes. The meanings of *m*, *n* and *w* have been explained in chapter 6 as well.

7.1 Pressure relief valve

The pressure relief valve used in this case study already had exceptionally high assembly efficiency (83.3%) and six components only. DFA analyses suggested that only one of the components was classified as "non-fully essential" and that it might benefit from redesign. On the other hand, the outstanding redesign (see Figure 7.5) made use of certain shape and material qualities of the remaining components, such as: the strength of the spring (column). Moreover, the stem was resized and redesigned to be fitted with a small guide that deflected the fluid flux and allow the spring to remain vertical during its operation. Thus, the redesign was found to be 100% efficient to assemble.

From this case study one can learn about optimal design configurations that exhibit minimum levels of complexity (see Comments 7.1.5.)

7.1.1 Original design of the pressure relief valve

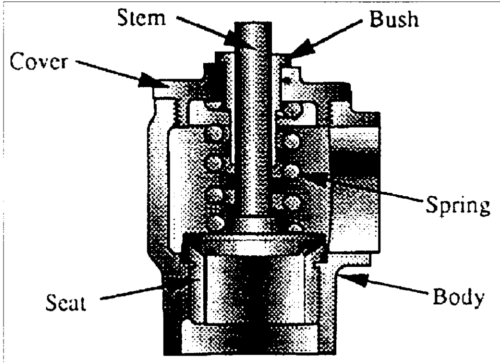
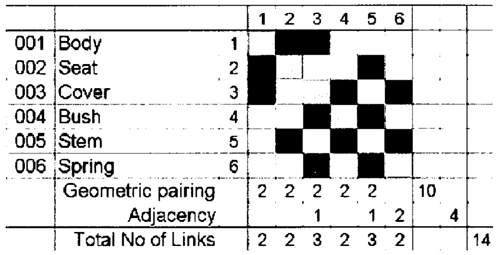


Figure 7.1 – Pressure relief valve (original design)

7.1.2 Matrix representation and graph representation



			1	2	3	4	5	6		
001	Body	1		■	■					
002	Seat	2	■				■			
003	Cover	3	■			■		■		
004	Bush	4			■		■			
005	Stem	5		■		■		■		
006	Spring	6			■		■			
	Geometric pairing		2	2	2	2	2		10	
	Adjacency				1		1	2	4	
	Total No of Links		2	2	3	2	3	2		14

Figure 7.2 – Matrix representation of pressure relief valve (original design)

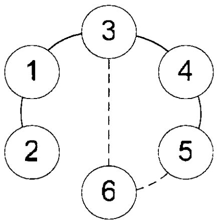


Figure 7.3 – Graph representation of pressure relief valve (original design)

7.1.3 Complexity evaluation

Number of components – n	6
Number of component types – m	6
Number of links – L	14
Connectivity ratio – (L/n)	2.33
Deviation from simple vertical assembly	16.7 %
Total cohesion or architectural compactness	46.7 %
Component variation factor – (m/n)	1.00
Interfaces	
Geometric interfaces	10
(Amount of geometric interfaces)	71 %
Adjacent interfaces	4
(Amount of adjacent interfaces)	29 %
Ratio (Adjacent/Geometric)	40 %
Interaction saturation	
Total number of actual interactions (LT)	7
Minimum number of interactions (LB)	5
Maximum number of possible interactions (UB)	15
Level of interaction saturation (LS)	20%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.165
Assembly complexity index with variant factor	1.165

7.1.4 Overall interface distribution graph

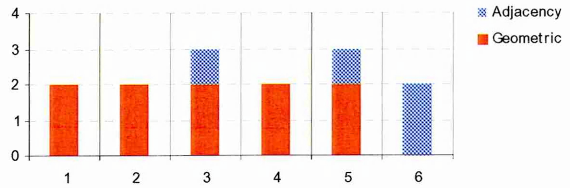


Figure 7.4 – Pressure relief valve (original design): Interface distribution graph

7.1.5 Comments

As mentioned during the introduction of this case study, it already has an outstanding DFA rating (83.3%). It is difficult to imagine that any further improvements could have been introduced. Nevertheless, this product presents a configuration that can provide valuable information to the complexity rating, especially due to its original high DFA rating.

- *Connectivity and vertical assemblability* – The connectivity ratio (2.33) is close the minimum level (2.00), hence it is understandable that the vertical assemblability has a high rating as well (86%).

- *Cohesion and Interaction Saturation* – The level of cohesion is close to the average 50%, which suggests that almost every component makes contact with another two components (vertical assembly). It does not present a distinctive hierarchical configuration (tree-like) –a ramification typical of complex systems. It can be seen that almost all components are "equally important", that is to say, there are sub-divisions or sub-systems that might be performing supporting tasks.
- *Component variation* – The number of components and types of components are exactly the same. There are not 'repeated' components, in this case, every component is inspected and analysed only once, and then it is assembled. No standardisation can be introduced. Therefore for a product with a 100% of variability, it can be said that its architecture is complex, but not complicated. There is no 'inflated' sense of complexity. In terms of product architecture, there is no complicatedness, which means that its complexity level is the minimum required for operation.
- *Interface variation* – The majority of the interfaces are of 'geometric type'. 71% of the interfaces are mainly geometric. Only one component presents a none geometric type of interface (Component 006 – spring). It suggests that geometric variation would not affect the shape of the rest of the components. Although, not entirely true, it does suggest that it can have a relatively different shape and the product would remain unchanged. The non-geometric interfaces displayed are not directly "mirrored" in their neighbouring components.

7.1.6 Suggested redesign of the pressure relief valve

The previous case study (original design) presented a high DFA rating that, in theory, suggested that no further improvements were required. However, the improvements suggested were introduced based on technical knowledge about components. In other works, physical characteristics (solid mechanics) of the components allowed a shift in the innovation of the product.

It is also interesting to see that the component that the component that allowed such a transformation, originally displayed non-geometric interfaces only.

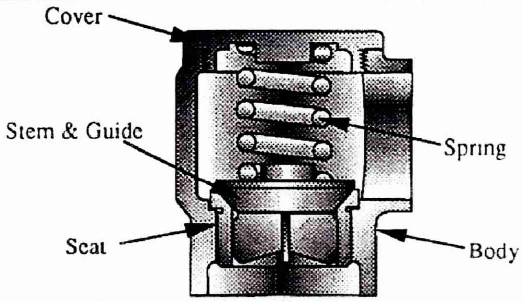
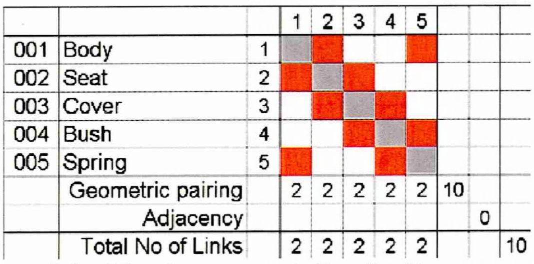


Figure 7.5 – Pressure relief valve (suggested redesign)

7.1.6.1 Matrix and graph representation



			1	2	3	4	5			
001	Body	1	▨	🟥			🟥			
002	Seat	2	🟥	▨	🟥					
003	Cover	3		🟥	▨	🟥				
004	Bush	4			🟥	▨	🟥			
005	Spring	5	🟥			🟥	▨			
	Geometric pairing		2	2	2	2	2	10		
	Adjacency								0	
	Total No of Links		2	2	2	2	2			10

Figure 7.6 – Matrix representation for Pressure relief valve (suggested redesign)

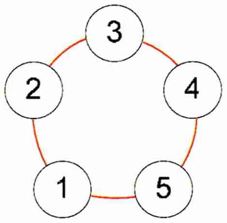


Figure 7.7 – Graph representation for pressure relief valve (suggested redesign)

7.1.6.2 Complexity evaluation

Number of components – n	5
Number of component types – m	5
Number of links – L	10
Connectivity ratio – (L/n)	2.00
Deviation from simple vertical assembly	0 %
Total cohesion or architectural compactness	50 %
Component variation factor – (m/n)	1.00
Interfaces	
Geometric interfaces	10
(Amount of geometric interfaces)	100 %
Adjacent interfaces	0
(Amount of adjacent interfaces)	0 %
Ratio (Adjacent/Geometric)	0 %
Interaction saturation	
Total number of actual interactions (LT)	5
Minimum number of interactions (LB)	4
Maximum number of possible interactions (UB)	10
Level of interaction saturation (LS)	16.7 %
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.00
Assembly complexity index with variant factor	1.00

7.1.6.3 Overall interface distribution graph

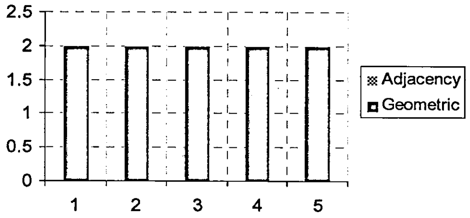


Figure 7.8 – Pressure valve (redesign) : Interface distribution graph

7.1.6.4 Comments

This elegant design does not only have the highest possible rating in design efficiency and assemblability, it also presents characteristics such as:

- *Connectivity and vertical assemblability* – This design presents a perfect connectivity ratio of 2.00, which ensures that every component "touches" two other components, subsequently exhibiting an impeccable vertical assemblability (zero deviation).
- *Cohesion and Interaction Saturation* – It also presents a well-balanced level of compactness.
- *Component variation* – As with the original design, it does not put on display any repeated components.
- *Interface variation* – Every component has exactly the same amount of interactions and interface type, therefore when its geometric shape is defined, all interfaces are immediately set on. Every component is now "geometrically aware" (or so to speak) of the presence of its neighbours.

This new design fully exploits all the technical understanding of the product. This knowledge derives from the physical characteristics of the components. However, as mentioned above, the designers analysed the component that, originally, displayed only non-geometric (adjacent) interfaces.

Furthermore, this product presents the minimum complexity level required to accomplish every single task.

7.2 Oil pump

The first case study presented a nearly perfect DFA score (both original and redesign even more so). Yet, products like these are rarely seen in everyday life. They do offer a clue as to what the baseline of complexity evaluation can be, but other products show some particular traits, more indicative of what complexity evaluation can point out.

Such is the case of this oil pump designed by a major UK vehicle manufacturer[34]. DFA analysis helped to reduced the part count from 32 to 12 (a 62.5% reduction) and the assembly cost by 79 %.

7.2.1 Original design

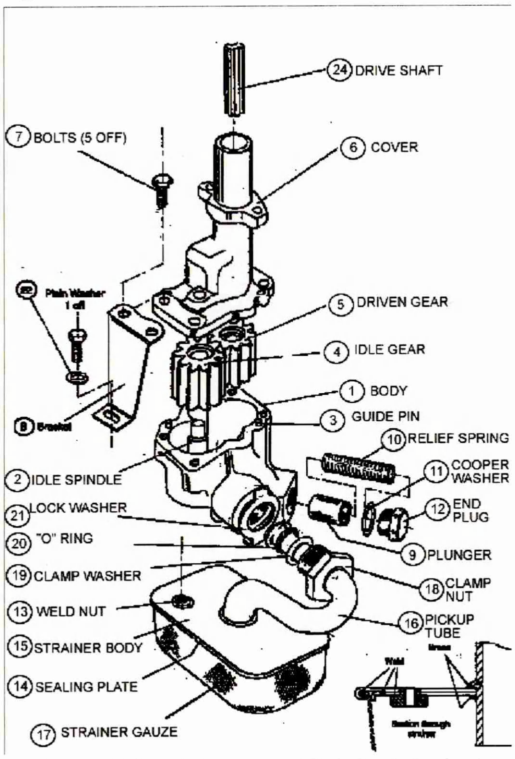


Figure 7.9 – Oil pump (Original design)

[34] Information about this case study was reproduced and copied with permission from TeamSET Concurrent Engineering tools. http://www.teamset.com/casestudy/oilpump.html

7.2.1.1 Matrix and graph representation

	Component		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25			
001	Body	1		■	■	■	■	■	■	■	■	■	■		■	■	■	■				■				■	■			
002	Idle spindler	2	■			■		■																						
003	Guiding pin	3	■					■																						
004	Idle gear	4	■	■			■	■																						
005	Driven gear	5	■			■		■																						
006	Cover	6	■	■	■	■	■		■	■	■	■	■	■																
07a	Bolt 01	7	■					■						■																
07b	Bolt 02	8	■					■						■																
07c	Bolt 03	9	■					■																						
07d	Bolt 04	10	■					■																						
07e	Bolt 05	11	■					■																						
008	Bracket	12						■	■	■																				
009	Plunger	13	■													■		■												
010	Relief spring	14	■												■			■												
011	Cooper washer	15	■															■												
012	End plug	16	■												■	■	■													
013	Weld nut	17																		■	■									
014	Sealing plate	18																	■		■	■								
015	Strainer body	19																	■	■		■	■							
016	Pickup tube	20	■																	■	■			■	■	■	■			
017	Strainer gauze	21																			■									
018	Clamp nut	22																				■				■				
019	Clamp washer	23																				■				■				
020	"O" Ring	24	■																			■		■	■					
021	Lock washer	25	■																			■								
	Geometric pairing		12	3	2	2	1	7	3	3	2	2	2	2	2	1		1	2	3	4	7	1	1	1	1	1	66		
	Adjacency		5		2	2	2	4						1	1	2	2	3						1	1	3	1		30	
	Total No of Links		17	3	4	4	3	11	3	3	2	2	2	3	3	3	2	4	2	3	4	7	1	2	2	4	2			96

Figure 7.10 – Matrix representation for Oil Pump (original design)

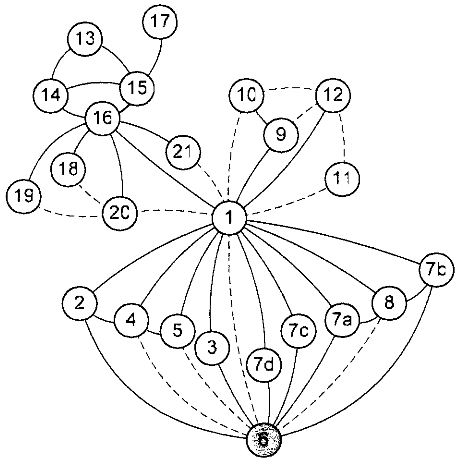


Figure 7.11 – Graph representation for Oil Pump (original design)

7.2.1.2 Complexity evaluation

Number of components – n	25
Number of component types – m	21
Number of links – L	96
Connectivity ratio – (L/n)	3.84
Deviation from simple vertical assembly	92 %
Total cohesion or architectural compactness	16 %
Component variation factor – (m/n)	0.84
Interfaces	
Geometric interfaces	66

(Amount of geometric interfaces)	69 %
Adjacent interfaces	30
(Amount of adjacent interfaces)	31 %
Ratio (Adjacent/Geometric)	45 %
Interaction saturation	
Total number of actual interactions (LT)	48
Minimum number of interactions (LB)	24
Maximum number of possible interactions (UB)	300
Level of interaction saturation (LS)	8.7 %
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.92
Assembly complexity index with variant factor	1.61

7.2.1.3 Overall interface distribution graph

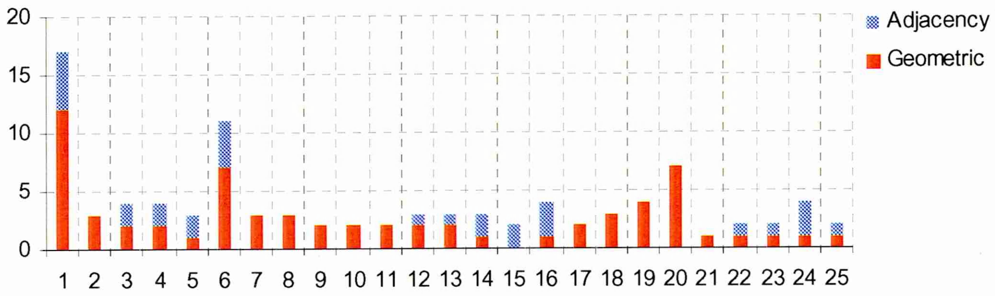


Figure 7.12 – Oil pump (original design): Overall interface distribution graph

7.2.1.4 Comments

This product shows that, as the number of parts increases, more information can be extracted from the product. In this case the characteristics of the product can be summarised as follows:

- *Connectivity and vertical assemblability* – It presents a high level of connectivity, which implies that every component influences on average every other *3.84* components. Obviously, this high number of interactions can be exhibited by products with "*localised interaction intricacy*". It can be seen that components 1 and 6 present up to 17 and 11 interactions, respectively. It is also reflected in the high deviation from vertical assemblability (92%). The 'overall distribution graph' (Figure 7.12) depicts the unbalance in the interface distribution (component 1, 9 and 20 have the highest number of interactions).

- *Cohesion and Interaction Saturation* – This product exhibits a rather "loose" architectural composition. The occurrence of loose configurations suggests the formation of sub-systems. As mentioned in the previous case study, this indicates that the product has complex characteristics.

  This value is offset in comparison to the number of interactions shown by three of the components. From the graph representation in Figure 7.11 it can be seen that clusters of components start to crop up [e.g. cluster (1, 9, 10, 11, 12) and cluster (13, 14, 15, 16, 17)], this has an impact on the cohesion level (sub-systems). Cluster formation lowers the overall cohesion level in a product, replacing it with *localised component interaction.* However, this localised interaction might imply that there are components (such as Component Nr 15, with only adjacent interactions and Nr. 21, with just one interaction) which will add up to the complicatedness of the design. Component 15 interacts with others, but it does not have a great impact on the shape, therefore it can be overseen during manufacturing. Component Nr. 21, being just a "satellite" means that its impact on the function of the product might be low (it might benefit from a combination with other components).

- *Component variation* – The component variation is still high (84%), however it just reflects the amount of components that have been *intentionally* described as similar (Bolt 01, Bolt 02 ...). The gears have been described as two separate, and therefore dissimilar units, but it they were to be taken as strictly as "geometrically similar" then the number of component types would drop to 20, and the component variation could be reduced to 80%. Although it is not a significant reduction, it exemplifies the need for a normalised method to characterise components (perhaps, through shape similarity classification? See chapter 8, *section: Shape Complexity Quantification*).

- *Interface variation* – Every component exhibits a geometric interface as the predominant type. However, there are certain components that do not have a strong influence on the geometric shape of its neighbouring elements; such is the case of Cooper washer – ID 15 (Component 011). It is clear that this component's shape depends on that of the connecting pipe, but it does not have a leading role, therefore its most predominant or identifiable interface type is merely the adjacency type. It can be seen that the "adjacent interface type" has a stronger presence in this product (up to 30% of the interfaces belong to this type). This type of interface is not usually considered in a component-oriented design. It also implies that such interfaces will not carry on "geometric information" downstream in the product development process. These components will not be aware of their neighbouring counterparts; this is bound to have an impact on the assembly process.

7.2.2 Oil pump – suggested redesign

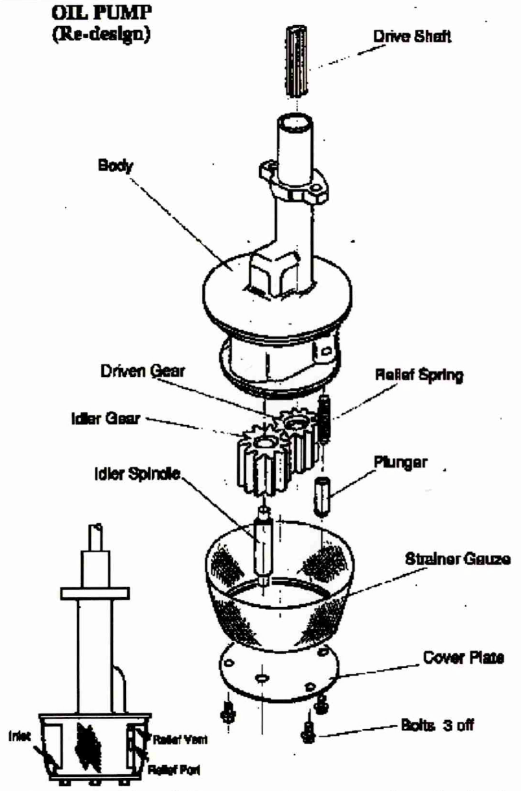


Figure 7.13 – Oil pump (suggested redesign)

7.2.2.1 Matrix and graph representation

Component		1	2	3	4	5	6	7	8	9	10	11			
Body	1		■	■	■	■	■	■	■	■	■	■			
Idle spindler	2	■		■					■						
Idle gear	3	■	■		■				■						
Driven gear	4	■		■					■						
Relief spring	5	■					■		■						
Plunger	6	■				■			■						
Strainer gauze	7	■							■						
Cover Plate	8	■	■	■	■	■	■	■		■	■	■			
Bolt 01	9	■							■						
Bolt 02	10	■							■						
Bolt 03	11	■							■						
Geometric pairing		8	3	2	1	1	2	2	5	2	2	2	30		
Adjacency		2		2	2	2	1		5					14	
Total No of Links		10	3	4	3	3	3	2	10	2	2	2			44

Figure 7.14 – Matrix representation for Oil pump (suggested redesign)

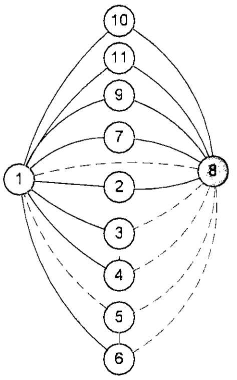


Figure 7.15 – Graph representation for Oil pump (suggested redesign)

7.2.2.2 Complexity evaluation

Number of components – n	11
Number of component types – m	9
Number of links – L	44
Connectivity ratio – (L/n)	4.00
Deviation from simple vertical assembly	100 %
Total cohesion or architectural compactness	40 %
Component variation factor – (m/n)	0.82
Interfaces	
Geometric interfaces	30
(Amount of geometric interfaces)	68.2%
Adjacent interfaces	14
(Amount of adjacent interfaces)	31.8%
Ratio (Adjacent/Geometric)	46.7%
Interaction saturation	
Total number of actual interactions (LT)	22
Minimum number of interactions (LB)	10
Maximum number of possible interactions (UB)	55
Level of interaction saturation (LS)	26.7%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	2.00
Assembly complexity index with variant factor	1.64

7.2.2.3 Overall interface distribution

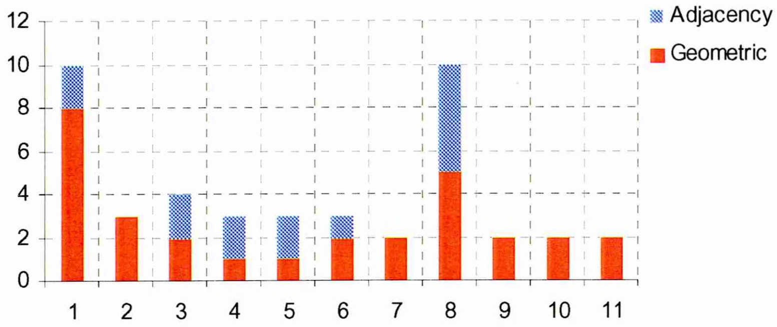


Figure 7.16 – Oil pump (redesign): Overall interface distribution

7.2.2.4 Comments

The number of components has been reduced by more than a half, which is bound to have an impact on the complexity of the product architecture. After all, complexity does depend directly on the number of components. Nevertheless, has the new design managed to eliminate any possible *complicatedness* as well? For that matter it is worth considering the same aspects as in previous case studies.

- *Connectivity and vertical assemblability* – Localised intricacy is still present, with components number 1 and 8 having the highest number of interactions. Numerically, it identifies the product as "hardly unlikely for vertical assembly," what it really reveals is a high connectivity that will be reflected in simultaneous insertions.

- *Cohesion and Interaction Saturation* – The cohesion has increased dramatically (more than twice than that of the original product), which means that the remaining components (or new components, for that matter) have managed to make a better use of the interactions available, reducing *architectural branch formation* (i.e. hierarchical configuration). Once again, numerically, it suggests that by making the product architecture more compact, the planning of assembly sequence has been simplified by reducing the number of possibilities. It can also be seen from its graph representation (Figure 7.15). Cohesion levels have increased, the product is more compact and the implications of modifying one component are more noticeable.

- *Component variation* – As in the original design, errors can be introduced by a misidentification of component types, for instance, components 3 and 4 (idler gear and driven gear, respectively) have originally been identified as two dissimilar types, but *how geometrically different are they?* From the assembly standpoint these two components are an identical set of gears. The same case can be argued for components 2 and 6 (idle spindle and plunger, respectively). In these circumstances, the actual number of different component types was recognised to be seven (7), as opposed to nine (9), as it was originally found. This new discovery brings the 'variant factor' from 0.82 to 0.64, which shows a decrease in 22%. This increase clearly affects the ACI. (Initial ACI = 1.64, actual ACI = 1.28). This new ACI immediately points out that the product has a less complex structure, hence less part-specific knowledge is required for its assembly. It also points out that it takes more time repeating certain operations (handling and insertion), which can be automated and therefore dealt with straightaway.

- *Interface variation* – The ratio of adjacent to geometric interfaces has remained almost constant (it just increased 1.7%), but the number of total interactions has been reduced by less than half (from 48 to 22). It, however, still shows a lack of geometric information being transferred from component to component.

This example shows that comparing the two designs does not give much useful information, why is that? Although both have the same functionality (to pump oil), the move from the original to the redesign has been so dramatic, that any understanding gained from a possible step-by-step analysis has been lost. Furthermore, the original and redesign product exemplify the differences and the impact of a proper (or improper) "component type classification."

7.3 *Staple remover*

A staple remover, similar to the ones used in almost every office, has also been analysed for its assemblability. The original design consists of 8 components, and its calculated manual assembly efficiency has been found to be 12.5% (only one of the claws was considered to be an essential part and the rest of components have been left to be either combined or eliminated altogether. The layout of the original design can be seen in Figure 7.17.

The proposed redesign makes use of the spring-like characteristics of most metals (in this case, Beryllium copper was suggested) and therefore the total number of components was reduced to just one. This dramatic result can be seen in Figure 7.21. Needless to say that the proposed design is the simplest of all.

7.3.1 Original design

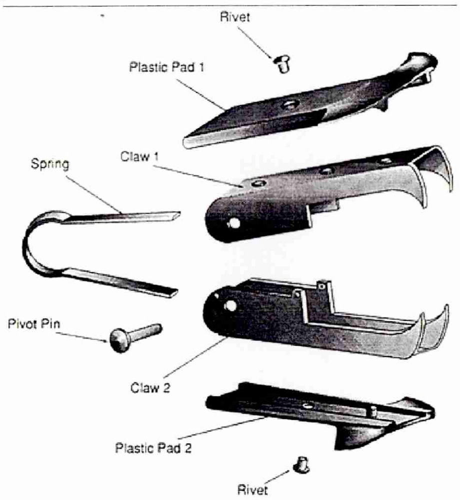
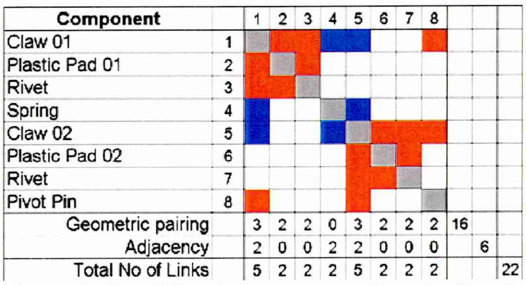


Figure 7.17 – Staple remover (original design)

7.3.1.1 Matrix and graph representation



Component		1	2	3	4	5	6	7	8			
Claw 01	1	▨	🟥	🟥	🟦	🟦			🟥			
Plastic Pad 01	2	🟥	▨	🟥								
Rivet	3	🟥	🟥	▨								
Spring	4	🟦			▨	🟦						
Claw 02	5	🟦			🟦	▨	🟥	🟥	🟥			
Plastic Pad 02	6					🟥	▨	🟥				
Rivet	7					🟥	🟥	▨				
Pivot Pin	8	🟥				🟥			▨			
Geometric pairing		3	2	2	0	3	2	2	2	16		
Adjacency		2	0	0	2	2	0	0	0		6	
Total No of Links		5	2	2	2	5	2	2	2			22

Figure 7.18 – Matrix representation for Staple remover (original design)

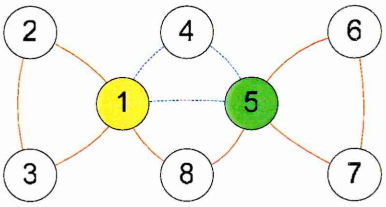


Figure 7.19 – Graph representation for Staple remover (original design)

7.3.1.2 Complexity evaluation

Number of components – n	8
Number of component types – m	5
Number of links – L	22
Connectivity ratio – (L/n)	2.75
Deviation from simple vertical assembly	37.5%
Total cohesion or architectural compactness	39.3%
Component variation factor – (n/m)	0.625
Interfaces	
Geometric interfaces	16
(Amount of geometric interfaces)	73%
Adjacent interfaces	6
(Amount of adjacent interfaces)	27%
Ratio (Adjacent/Geometric)	38%
Interaction saturation	
Total number of actual interactions (LT)	11
Minimum number of interactions (LB)	7
Maximum number of possible interactions (UB)	28
Level of interaction saturation (LS)	19.0%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.375
Assembly complexity index with variant factor	0.859

7.3.1.3 Overall interface distribution

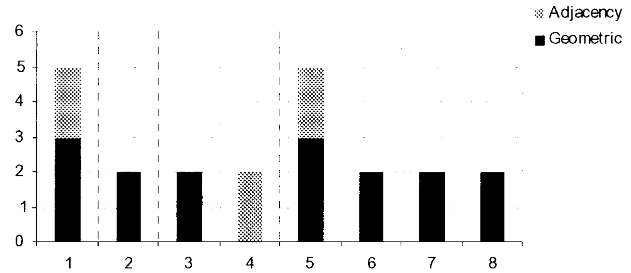


Figure 7.20 – Staple remover (original design): Overall interface distribution

7.3.1.4 Comments

This product does not present a large level of assembly difficulty. It has been selected for re-design due of its low design efficiency, and almost half of the product is found to be redundant. Its complexity level is minimal, yet much assembly time is spent on repeated components (adding up to complicating the assembly). How does it behave in comparison with the other factors selected?

- *Connectivity and vertical assemblability* – As mentioned above, its assembly efficiency is not affected by the number of repeated components, its connectivity ratio is 2.75 (and a deviation of 37.5%) it only suggests that some non-vertical-assembly operations need to be performed.
- *Cohesion and Interaction Saturation* – Its level of cohesion is still high (slightly higher than 30%), which can be appreciated in Figure 7.19, with an almost perfect symmetry. It is extremely well balanced.
- *Component variation* – Variation factor is significantly lower than 80% (actually, 37.5% of its components are not repeated). What is its significance? It means that once around 60% of the pieces have been studied, the rest of the time is "wasted" in repeated operations. On the other hand, much can be learnt from a component with such low level of component variance. It really proves to be a simple object [ACI(unit-weighted) = 0.859], but it is "perceived" as higher due to the amount of components that are repeated. It raises the alarm in terms of "intended repeatability", was it the intention of the designer?
- *Interface variation* – The dominant type of interaction is geometric, by far. This ensures that proper geometric information is transmitted downstream.

If everything seems to be fine with this product, then why was it redesigned? Well, it is worth noting that the ACI (Assembly Complexity Index) shown in the complexity summary above does not contain the weight of the interfaces (no geometric evaluation has been performed). Perhaps more importantly, in terms of product structure, is that this design has a high number of components to perform its ultimate task: removing a staple. Many of the components perform supportive tasks.

7.3.2 Staple remover – suggested redesign

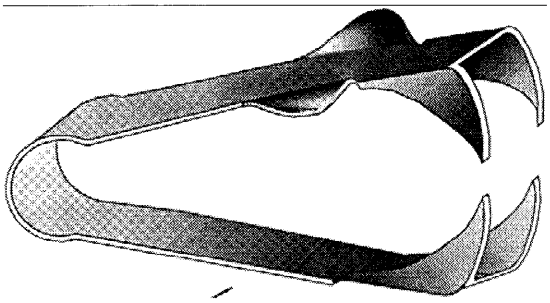


Figure 7.21 – Staple remover (suggested redesign)

7.3.2.1 Comments

There is not much that one can say about this product, it has been simplified to the maximum, with no interacting parts, there is no assembly process.

7.4 Motor drive assembly

A motor drive assembly[35] is required to sense and control its position on two steel guide rails. The motor must be fully enclosed for aesthetic reasons and have a removable cover for access to adjustment of the position sensor. The principal requirements are a rigid based designed to slide up and down the guide rails which will both support the motor and locate the sensor.

One initial design (Figure 7.22) consists of an assembly with 19 pieces. Its design efficiency using the Boothroyd DFA methodology was calculated to be 7.5 %.

The above mentioned design was later revised. The suggested redesign (Figure 7.26) produced improvement its assemblability. This new solution has seven components and a design efficiency value of 26%, once again this design efficiency was calculated using the Boothroyd DFA methodology.

7.4.1 Original design

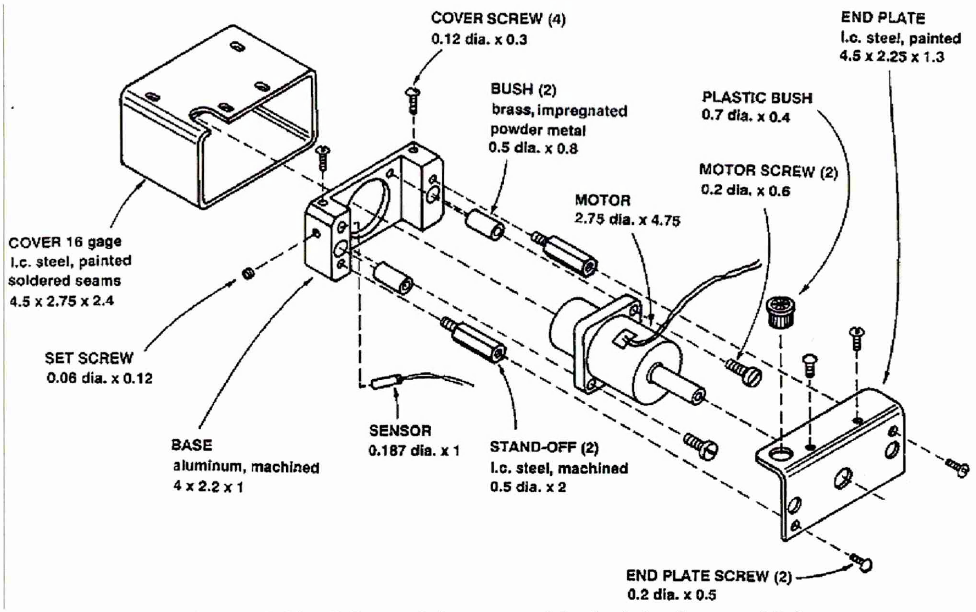
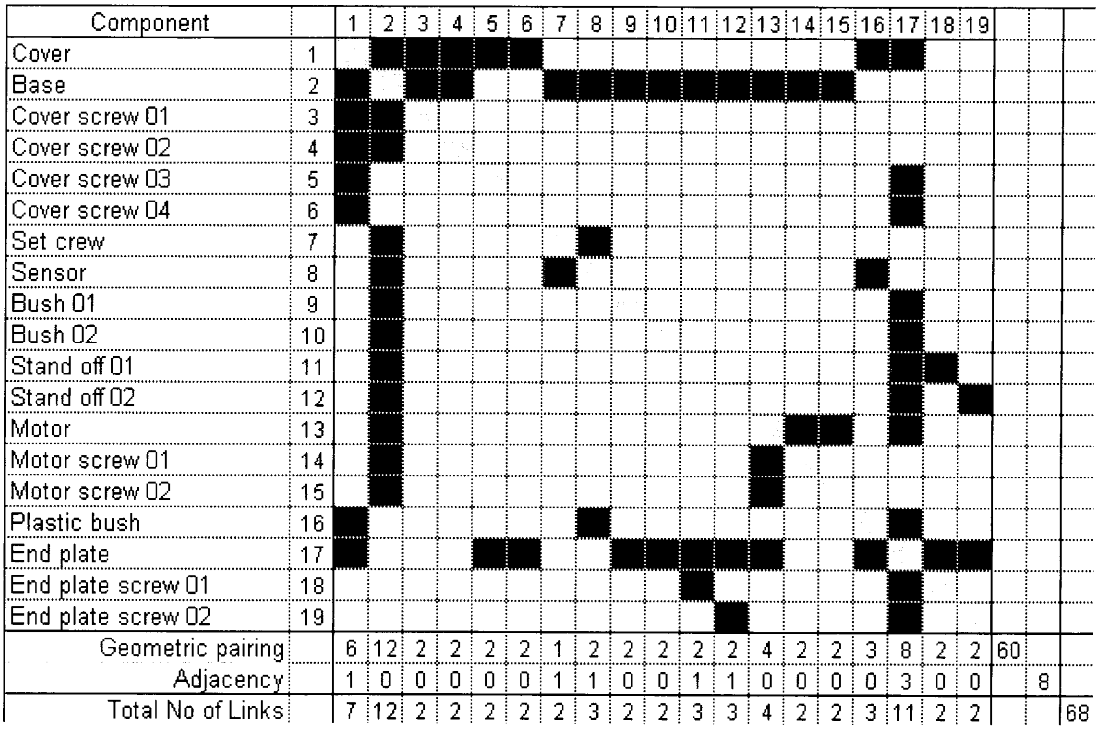


Figure 7.22 – Motor drive assembly (original assembly)

[35] Information for this case study was reproduced from Boothroyd, G., Dewhurst, P., and Knight, W.A. (1994). "Product design for manufacture and assembly." New York: M. Dekker; ISBN: 0824791762.

7.4.1.1 Matrix and graph representation



Component		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19			
Cover	1		■	■	■	■	■										■	■					
Base	2	■		■	■			■	■	■	■	■	■	■	■	■							
Cover screw 01	3	■	■																				
Cover screw 02	4	■	■																				
Cover screw 03	5	■																■					
Cover screw 04	6	■																■					
Set crew	7		■						■														
Sensor	8		■					■									■						
Bush 01	9		■															■					
Bush 02	10		■															■					
Stand off 01	11		■															■	■				
Stand off 02	12		■															■		■			
Motor	13		■												■	■		■					
Motor screw 01	14		■											■									
Motor screw 02	15		■											■									
Plastic bush	16	■							■									■					
End plate	17	■				■	■			■	■	■	■	■			■		■	■			
End plate screw 01	18											■						■					
End plate screw 02	19												■					■					
Geometric pairing		6	12	2	2	2	2	1	2	2	2	2	2	4	2	2	3	8	2	2	60		
Adjacency		1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	3	0	0		8	
Total No of Links		7	12	2	2	2	2	2	3	2	2	3	3	4	2	2	3	11	2	2			68

Figure 7.23 – Matrix representation for Motor drive assembly (original design)

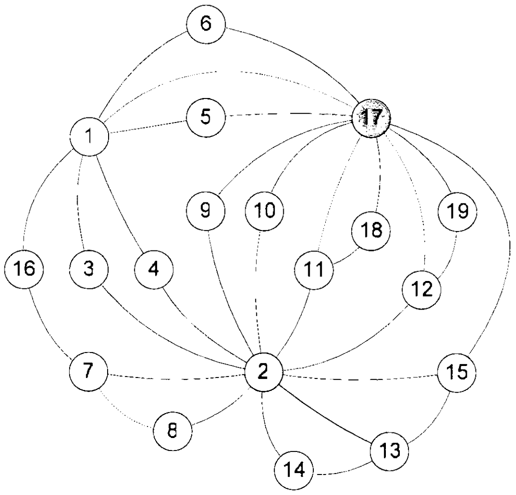


Figure 7.24 – Graph representation for Motor drive assembly (original design)

7.4.1.2 Complexity evaluation

Number of components – n	19
Number of component types – m	13
Number of links – L	68
Connectivity ratio – (L/n)	3.58
Deviation from simple vertical assembly	78.9%
Total cohesion or architectural compactness	19.9%
Component variation factor – (n/m)	0.68
Interfaces	
Geometric interfaces	60

(Amount of geometric interfaces)	88.2%
Adjacent interfaces	8
(Amount of adjacent interfaces)	11.8%
Ratio (Adjacent/Geometric)	13.3%
Interaction saturation	
Total number of actual interactions (LT)	34
Minimum number of interactions (LB)	18
Maximum number of possible interactions (UB)	171
Level of interaction saturation (LS)	10.5%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.79
Assembly complexity index with variant factor	1.22

7.4.1.3 Overall interface distribution

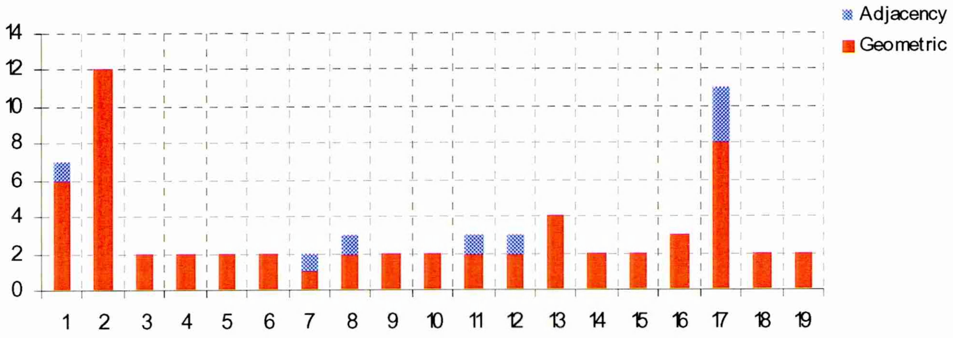


Figure 7.25 – Motor drive (original design): Overall interface distribution.

7.4.1.4 Comments

It is immediately recognised that this product has a high number of bolts (comparatively). This characteristic is bound to produce an inefficient design from the DFA standpoint. In these circumstances it will ultimately skew the complexity index by producing a low component variation factor. How the rest of the factors behave in this particular composition can be seen as follows.

- *Connectivity and vertical assemblability* – Its high connectivity and high deviation from simple vertical assembly clearly suggests that numerous operations are happening at the same time (multiple and simultaneous insertions as well as lateral insertion). Two components dominate the interactivity in the product, drawing all the resources towards them (i.e. Component nr 2 and 17). The first component (nr 2) with high number of connections is the base, and as expected, this high connectivity is understood, however the second one is the end plate. Does it really require such high number of connections? It is clearly noted that use of bolts has created this increase in connectivity; therefore they have introduced complicatedness to the structure.

- *Cohesion and Interaction Saturation* – It presents a very low level of cohesion. As seen in previous cases it means hierarchical configuration or tree-like branching.
- *Component variation* – It presents a low Component variation, mainly due to fastening components. Much of the time is spent in dealing with *repeated components*, have the designers made them similar purposely? It could be the case, had majority not been fastening parts. Complexity is certainly not an issue, because specific component information is gathered quickly, but the low number of dissimilar components types does suggest an inclination to high complicatedness.
- *Interface variation* – No particular comments.

7.4.2 Motor drive – suggested redesign

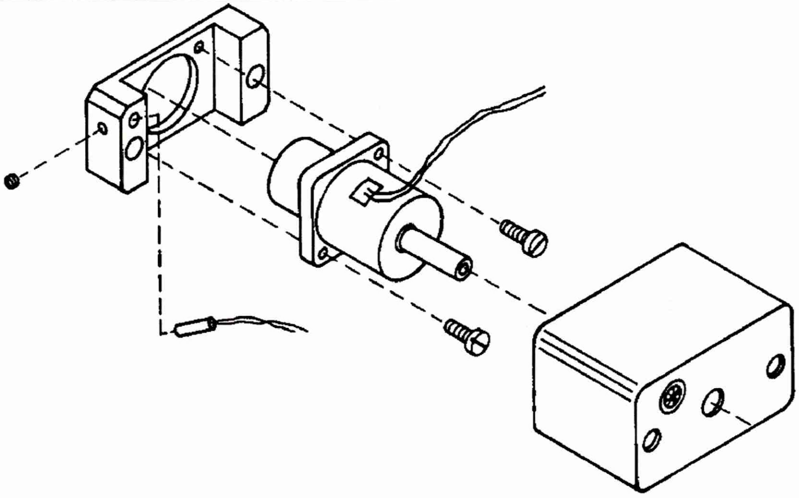


Figure 7.26 – Motor drive assembly (suggested redesign)

7.4.2.1 Matrix and graph representation

Component		1	2	3	4	5	6	7			
Base	1	▒	🟥	🟥	🟥	🟥	🟥	🟥			
Set screw	2	🟥	▒	🟦							
Sensor	3	🟥	🟦	▒				🟥			
Motor	4	🟥			▒	🟥	🟥	🟥			
Motor screw 01	5	🟥			🟥	▒					
Motor screw 02	6	🟥			🟥		▒				
Cover	7			🟥	🟥			▒			
Geometric pairing		6	1	2	4	2	2	3	20		
Adjacency		0	1	1	0	0	0	0		2	
Total No of Links		6	2	3	4	2	2	3			22

Figure 7.27 – Matrix representation for Motor drive assembly (suggested redesign).

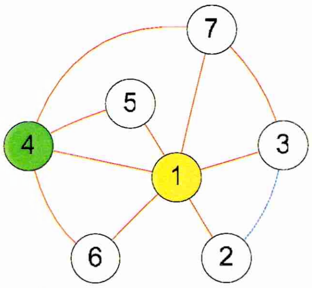


Figure 7.28 – Graph representation for motor drive assembly (suggested redesign)

7.4.2.2 Complexity evaluation

Number of components – n	7
Number of component types – m	6
Number of links – L	22
Connectivity ratio – (L/n)	3.14
Deviation from simple vertical assembly	57.1%
Total cohesion or architectural compactness	52.4%
Component variation factor – (m/n)	0.86
Interfaces	22
Geometric interfaces	20
(Amount of geometric interfaces)	90.9%
Adjacent interfaces	2
(Amount of adjacent interfaces)	9.1%
Ratio (Adjacent/Geometric)	10.0%
Interaction saturation	
Total number of actual interactions (LT)	11

Minimum number of interactions (LB)	6
Maximum number of possible interactions (UB)	21
Level of interaction saturation (LS)	33.3 %
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.57
Assembly complexity index with variant factor	1.35

7.4.2.3 Overall interface distribution

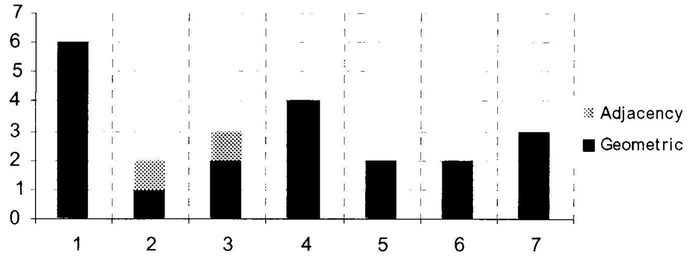


Figure 7.29 – Motor drive (suggested redesign): Overall interface distribution.

7.4.2.4 Comments

- *Connectivity and vertical assemblability* – Simultaneous and non-vertical assembly operations.
- *Cohesion and Interaction Saturation* – It has improved considerably. It is higher than the top average of 50%. The interaction saturation has increased threefold.
- *Component variation* – It is affected by the incorporation of two screws. Otherwise, it has a high level of variety.
- *Interface variation* – It presents a high geometrical awareness (91 %). The only non-geometric interface is required due to the relative movement between the "set screw" and "the sensor".

7.5 Wiper motor

The 20W rear screen wiper motor is intended for use in all market zones and is to operate with either 12V or 24V electrical systems. Some of the technical specifications of this product are:

- *Maintenance* – the motor assembly may be considered as an exchange unit and therefore does not require any special service or repair.
- *Physical size* – the external dimensions of the unit should comply within an envelope of maximum length of 140 mm and maximum cylindrical diameter of 80 mm.
- *Other specifications* – for power requirement a flux path must be provided between the magnetic poles. Brushes must be opposed at 180 degrees on the commutator and they must be fixed in a positive relationship with the magnetic poles.

The original design (Figure 7.30) required a total of 27 parts, problems with tolerance build-up requires self aligning bushes and there was a need for a brush hold back tool. Moreover, the calculated assembly efficiency was found to be just 19.3%.

The suggested redesign reduced the part count down to 6 components only. Combination of components produced a one-piece formed can and a one-piece moulded end cap and bearing. Other improvements were the bush alignment on assembly with can, an all in line assembly. Snap held brush spring and connectors were combined and face commutator preloaded brushes on assembly.

7.5.1 Original design

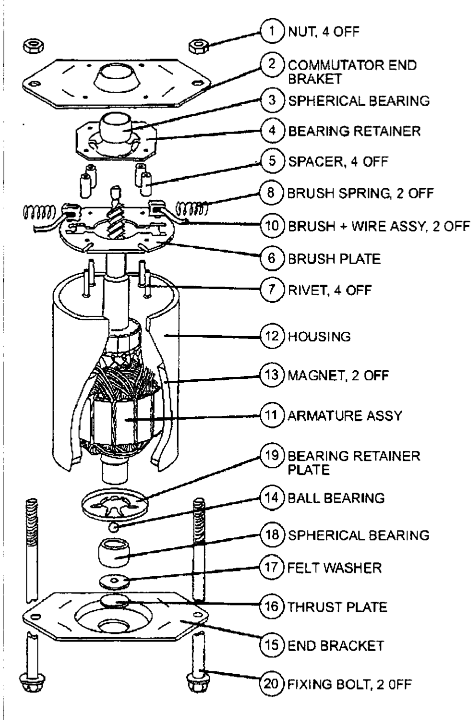


Figure 7.30 – Wiper motor (Original design).

7.5.1.1 Matrix and graph representation

	Components		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
1A	Nut	1			■																											■				
1B	Nut	2			■																												■			
02	Commutator end bracket	3	■	■		■							■	■	■	■							■									■	■			
03	Spherical bearing	4			■		■															■														
04	Bearing retainer	5				■		■	■	■	■		■	■	■	■						■														
5A	Spacer	6					■					■	■																							
5B	Spacer	7					■					■		■																						
5C	Spacer	8					■					■			■																					
5D	Spacer	9					■					■				■																				
06	Brush plate	10						■	■	■	■		■	■	■	■			■	■	■															
7A	Rivet	11			■		■	■				■																								
7B	Rivet	12			■		■		■			■																								
7C	Rivet	13			■		■			■		■																								
7D	Rivet	14			■		■				■	■																								
8A	Brush spring	15																		■			■													
8B	Brush spring	16																			■		■													
09	Rubber grommet	17										■								■	■															
10A	Brush and wire assembly	18										■					■		■			■														
10B	Brush and wire assembly	19										■						■	■			■														
11	Armature assembly	20				■	■													■	■					■				■	■					
12	Housing	21			■												■	■						■	■		■									
13A	Magnet	22																					■													
13B	Magnet	23																					■													
14	Ball bearing	24																				■							■							
15	End bracket	25																					■					■				■	■			
16	Thrust plate	26																									■									
17	Felt washer	27																								■				■						
18	Spherical bearing	28																				■							■		■					
19	Bearing retainer plate	29																				■								■						
20A	Fixing bolt	30	■		■																						■									
20B	Fixing bolt	31		■	■																						■									
	Geometric pairing		1	1	7	3	6	1	1	1	1	6	4	4	4	4	0	0	2	2	2	5	2	1	1	2	3	1	2	3	2	3	3	78		
	Adjacency		1	1	3	0	4	2	2	2	2	5	0	0	0	0	2	2	1	2	2	2	4				1								38	
	Total No of Links		2	2	10	3	10	3	3	3	3	11	4	4	4	4	2	2	3	4	4	7	6	1	1	2	4	1	2	3	2	3	3			116

Figure 7.31 – Matrix representation for wiper motor (original design)

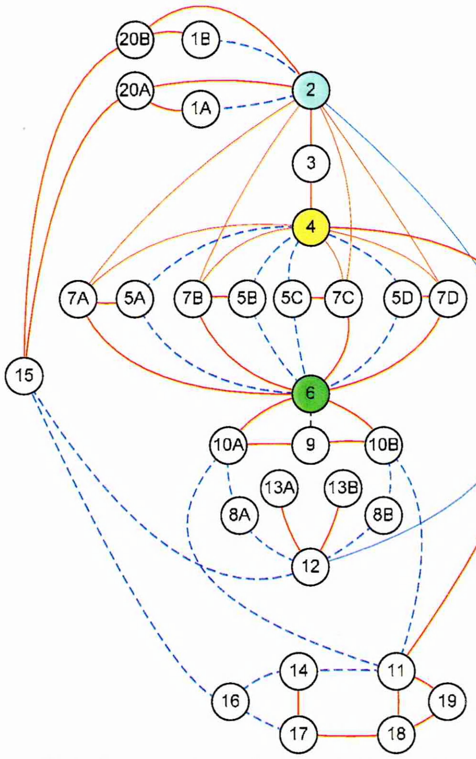


Figure 7.32 – Graph representation for wiper motor (original design).

7.5.1.2 Complexity evaluation

Number of components – n	31
Number of component types – m	19
Number of links – L	116
Connectivity ratio – (L/n)	3.74
Deviation from simple vertical assembly	87.1%
Total cohesion or architectural compactness	12.5%
Component variation factor – (m/n)	0.61
Interfaces	
Geometric interfaces	78
(Amount of geometric interfaces)	67.2%
Adjacent interfaces	38
(Amount of adjacent interfaces)	32.8%
Ratio (Adjacent/Geometric)	48.7%
Interaction saturation	
Total number of actual interactions (LT)	58
Minimum number of interactions (LB)	30
Maximum number of possible interactions (UB)	465
Level of interaction saturation (LS)	6.4%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.87
Assembly complexity index with variant factor	1.14

7.5.1.3 Overall interface distribution

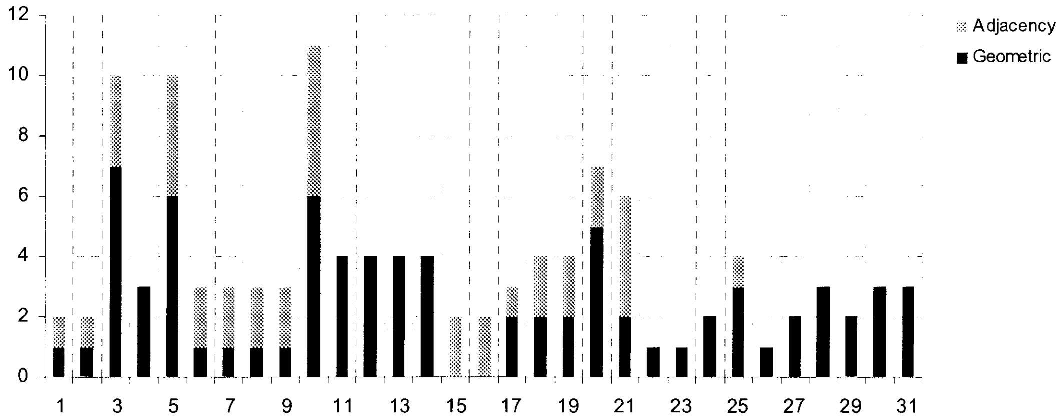


Figure 7.33 – Wiper motor (original design): Overall interface distribution.

7.5.1.4 Matrix representation – reshuffled banded matrix

			1A	1B	3	4	2	7A	7B	7C	7D	5A	5B	5C	5D	6	9	10A	10B	8A	8B	13A	13B	12	19	11	14	18	17	16	15	20A	20B			
Component			1	2	4	5	3	11	12	13	14	6	7	8	9	10	17	18	19	15	16	22	23	21	29	20	24	28	27	26	25	30	31			
Nut	1A	1					■																									■				
Nut	1B	2					■																										■			
Spherical bearing	03	4				■	■																													
Bearing retainer	04	5			■			■	■	■	■	■	■	■	■											■										
Commutator end bracket	02	3	■	■	■			■	■	■	■													■								■	■			
Rivet	7A	11				■	■					■				■																				
Rivet	7B	12				■	■						■			■																				
Rivet	7C	13				■	■							■		■																				
Rivet	7D	14				■	■								■	■																				
Spacer	5A	6				■		■								■																				
Spacer	5B	7				■			■							■																				
Spacer	5C	8				■				■						■																				
Spacer	5D	9				■					■					■																				
Brush plate	06	10						■	■	■	■	■	■	■	■		■	■	■																	
Rubber grommet	09	17														■		■	■																	
Brush and wire assembly	10A	18														■	■			■						■										
Brush and wire assembly	10B	19														■	■				■					■										
Brush spring	8A	15																■						■												
Brush spring	8B	16																	■					■												
Magnet	13A	22																						■												
Magnet	13B	23																						■												
Housing	12	21					■													■	■	■	■								■					
Bearing retainer plate	19	29																								■		■								
Armature assembly	11	20				■												■	■						■		■	■								
Ball bearing	14	24																								■			■							
Spherical bearing	18	28																							■	■			■							
Felt washer	17	27																									■	■		■						
Thrust plate	16	26																											■		■					
End bracket	15	25																						■						■		■	■			
Fixing bolt	20A	30	■				■																								■					
Fixing bolt	20B	31		■			■																								■					
Geometric pairing			1	1	2	6	7	4	4	4	4	1	1	1	1	6	2	2	2	0	0	1	1	2	2	3	1	3	2	0	2	3	3	72		
Adjacency			1	1	0	4	3	0	0	0	0	2	2	2	2	5	1	2	2	2	2	0	0	4	0	3	1	0	1	2	2	0	0		44	
Total No of Links			2	2	2	10	10	4	4	4	4	3	3	3	3	11	3	4	4	2	2	1	1	6	2	6	2	3	3	2	4	3	3			116

Figure 7.34 – Matrix representation for Wiper motor (original design) - Banded matrix.

7.5.1.5 Comments

For a start, this product clearly exemplifies the additional benefits of using the adjacency matrix, component clustering. This 'by-product' allows cluster to be identified for parallel assembly or modular architecture (see Figure 7.34, and chapter 6 for more information on that matter). On the other hand, this product presents a great deal of component repetition, hence the increase in complicatedness or inflated complexity report.

- *Connectivity and vertical assemblablity* – The connectivity level is above the accepted value (higher than 2.5). Turnover, simultaneous insertions and other operations affect the vertical assembablity, hence the high deviation percentage.
- *Cohesion and Interaction Saturation* – Loose cohesion. Tree-like configuration. Localised interaction. Is this required? Are these components performing essential functions?
- *Component variation* – There are lots of repeated components. Arrays and sets of components abound. Component variation is very low.
- *Interface variation* – There is a high level of non-geometric interfaces (adjacency type). Geometric information is not carried on downstream.

This product presents one of the "by-products" of using the "adjacency matrix evaluation": *component clustering identification.* Figure 7.34 exemplifies what can be achieved just by shuffling rows and columns until collections of adjacent components, with multiple interactions amongst them are identified.

7.5.2 Wiper motor – suggested redesign

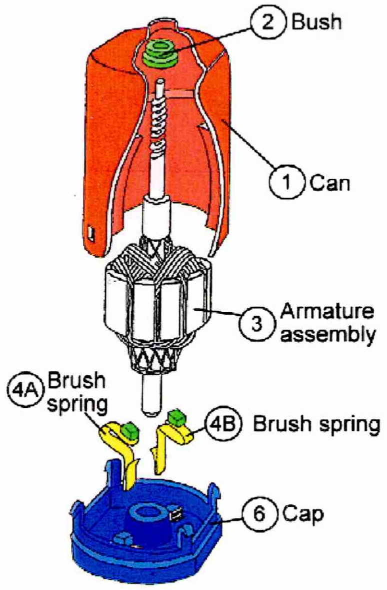


Figure 7.35 – Wiper motor suggested redesign.

7.5.2.1 Matrix and graph representation

	Components		1	2	3	4	5	6		
01	Can	1		■				■		
02	Bush	2	■		■					
03	Armature assembly	3		■		■	■	■		
4A	Brush spring	4			■			■		
4B	Brush spring	5			■			■		
05	Cap	6	■		■	■	■			
	Geometric pairing		2	2	2	1	1	4	12	
	Adjacency		0	0	2	1	1	0	4	
	Total No of Links		2	2	4	2	2	4		16

Figure 7.36 – Matrix representation for wiper motor (suggested redesign).

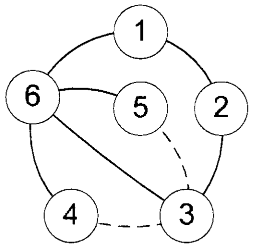


Figure 7.37 – Graph representation for wiper motor (suggested redesign).

7.5.2.2 Complexity evaluation

Number of components – n	6
Number of component types – m	5
Number of links – L	16
Connectivity ratio – (L/n)	2.67
Deviation from simple vertical assembly	33.3%
Total cohesion or architectural compactness	53.3%
Component variation factor – (m/n)	0.83
Interfaces	16
Geometric interfaces	12
(Amount of geometric interfaces)	75.0%
Adjacent interfaces	4
(Amount of adjacent interfaces)	25.0%
Ratio (Adjacent/Geometric)	33.3%
Interaction saturation	
Total number of actual interactions (LT)	8
Minimum number of interactions (LB)	5
Maximum number of possible interactions (UB)	15
Level of interaction saturation (LS)	30.0%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.335
Assembly complexity index with variant factor	1.108

7.5.2.3 Overall interface distribution

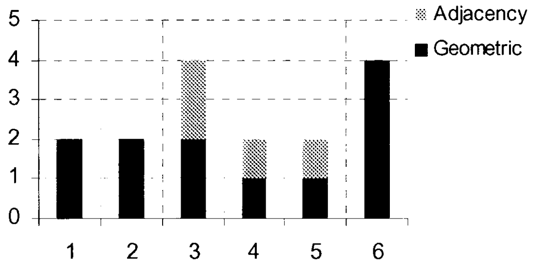


Figure 7.38 – Wiper motor (suggested redesign): Overall interface distribution.

7.5.2.4 Comments

- *Connectivity and vertical assemblability* – The deviation from vertical assembly operations is minimal and within the accepted range (approximately 30%)
- *Cohesion and Interaction Saturation* – The cohesion level has been dramatically improved, from 12.5% to 53.3%. Even higher than the average top maximum of 50%. The saturation level also been increased from 6.4% to 30%. The product is therefore more compact. Every component is fulfilling more specific tasks (perhaps more essential?).
- *Component variation* – The repetition of components has been dramatically cut down. It has an impact on the assembly operations, every part needs to be inspected and no knowledge is stored for future operations. However, much is saved in time. Only one component has been repeated, but this is for operational purposes.
- *Interface variation* – The geometric awareness has increased. The sole remaining non-geometric interface is for operational purposes.

7.6 *Door latch mechanism*

The double-action latch mechanism provides easy operator access to the paper transports to clear paper jams and to the developer sump area to load dry ink. The door latch mechanism also provides a key lock assembly which prevents unauthorized entry into these areas of the machine. The layout of the original door latch mechanism is shown in Figure 7.1. The original mechanism consists of 62 components, which require an estimated assembly time of 6.9 minutes. The manual assembly efficiency was calculated to be just 4.8%.

After a DFA analysis, there is a suggested redesign of the door latch mechanism, which has a total number of 17 components and an estimated assembly time of 1.48 minutes, which gives it a calculated manual assembly efficiency of 22.5%.

7.6.1 Original design

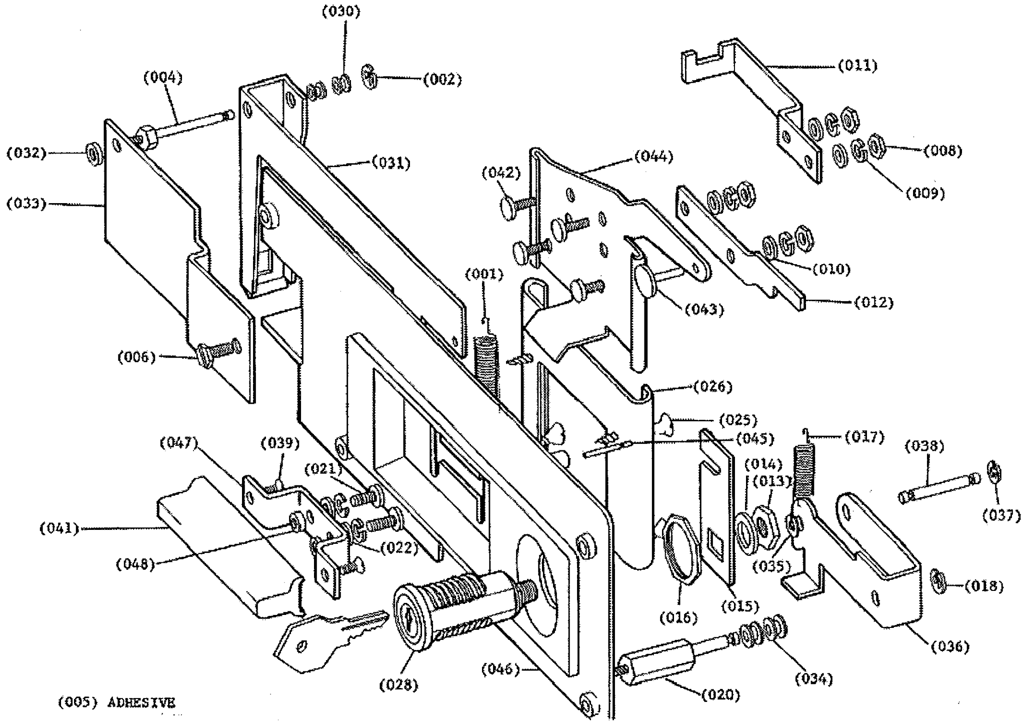


Figure 7.39 – Door latch original design.

7.6.1.1 Matrix and graph representation

Component	ID		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21			
Bracket release	004	1	▨				🟥								🟥						🟥			2	1	3
Circlip	014	2		▨						🟥	🟦													1	1	2
Clip handle	009	3			▨	🟥		🟦					🟦											1	2	3
Handle	017	4			🟥	▨	🟥	🟥																3		3
Housing	015	5	🟥			🟥	▨			🟥				🟦	🟥						🟥			5	1	6
Inner guide	010	6			🟦	🟥		▨			🟥		🟥											3	1	4
Latch arm	006	7							▨					🟥		🟦						🟥		2	1	3
Lock	016	8		🟥			🟥			▨	🟥	🟥											🟥	5		5
Lock arm	008	9		🟦				🟥		🟥	▨												🟦	2	2	4
Nut	007	10								🟥		▨											🟦	1	1	2
Outer guide	013	11			🟦			🟥					▨				🟥	🟥	🟥	🟥				5	1	6
Plate	-	12					🟦		🟥					▨		🟥								2	1	3
Push clip 01	003	13	🟦				🟥								▨									1	1	2
Push clip 02	005	14							🟦					🟥		▨								1	1	2
Screws 01	012	15											🟥				▨							1		1
Screws 02	012	16											🟥					▨						1		1
Screws 03	-	17											🟥						▨					1		1
Screws 04	-	18											🟥							▨				1		1
Spring 01	001	19	🟥				🟥														▨			2		2
Spring 02	002	20							🟥													▨		1		1
Washer	-	21								🟥	🟦	🟦											▨	1	2	3
Geometric pairing			2	1	1	3	5	3	2	5	2	1	5	2	1	1	1	1	1	1	2	1	1	42		
Adjacency			1	1	2	0	1	1	1	0	2	1	1	1	1	1	0	0	0	0	0	0	2		16	
Total No Interactions			3	2	3	3	6	4	3	5	4	2	6	3	2	2	1	1	1	1	2	1	3			58

Figure 7.40 – Matrix representation for Door latch (original design).

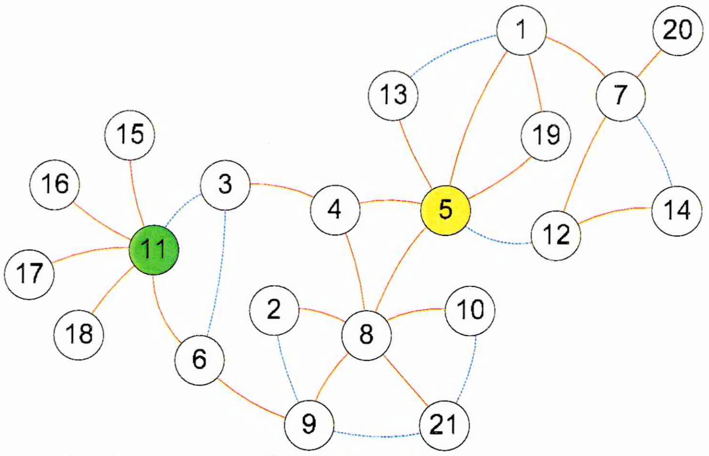


Figure 7.41 – Graph representation for Door latch mechanism (original design).

7.6.1.2 Complexity evaluation

Number of components – n	21
Number of component types – m	16
Number of links – L	58
Connectivity ratio – (L/n)	2.76
Deviation from simple vertical assembly	38.1%
Total cohesion or architectural compactness	13.8%
Component variation factor – (m/n)	0.76

Interfaces	58
Geometric interfaces	42
(Amount of geometric interfaces)	72.4%
Adjacent interfaces	16
(Amount of adjacent interfaces)	27.6%
Ratio (Adjacent/Geometric)	38.1%
Interaction saturation	
Total number of actual interactions (LT)	29
Minimum number of interactions (LB)	20
Maximum number of possible interactions (UB)	210
Level of interaction saturation (LS)	4.7%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.38
Assembly complexity index with variant factor	1.05

7.6.1.3 Overall interface distribution

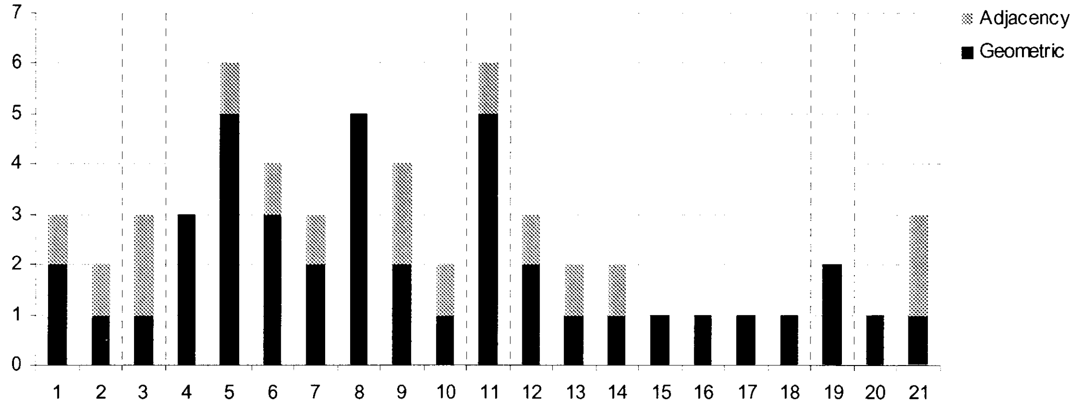


Figure 7.42 – Door latch (Original design): Overall interface distribution.

7.6.1.4 Comments

- *Connectivity and vertical assemblability* – Surprisingly the connectivity and vertical assemblability levels are within accepted values. Therefore this metric does not reveal much about the nature of the product.

- *Cohesion and Interaction Saturation* – The cohesion level is low, suggesting a ramification in its configuration. The interaction saturation is extremely low. This low utilisation of all the possible connections available implies that the product is divided into sub-assemblies, perhaps some components are just used to support other components and they do not contribute to the overall functionality required of the product.

- *Component variation* – The large number of fastening components has an impact on the variety in types of components, with the implications stated in previous case studies. As a consequence, there is a high possibility for sheer complicatedness.
- *Interface variation* – There is quite a significant number of non-geometric interfaces, especially due to fastening components.

7.6.1.5 Matrix representation – reshuffled banded matrix

Component	ID		14	12	7	20	5	1	19	13	8	2	9	21	10	4	3	6	11	15	16	17	18			
Push clip 02	005	14	▨	🟥	🟦																			1	1	2
Plate	-	12	🟥	▨	🟥		🟦																	2	1	3
Latch arm	006	7	🟦	🟥	▨	🟥																		2	1	3
Spring 02	002	20			🟥	▨																		1		1
Housing	015	5		🟦			▨	🟥	🟥	🟥	🟥					🟥								5	1	6
Bracket release	004	1					🟥	▨	🟥	🟦														2	1	3
Spring 01	001	19					🟥	🟥	▨															2		2
Push clip 01	003	13					🟥	🟦		▨														1	1	2
Lock	016	8					🟥				▨	🟥	🟥	🟥	🟥									5		5
Circlip	014	2									🟥	▨	🟦											1	1	2
Lock arm	008	9									🟥	🟦	▨	🟦				🟥						2	2	4
Washer	-	21									🟥		🟦	▨	🟦									1	2	3
Nut	007	10									🟥			🟦	▨									1	1	2
Handle	017	4					🟥									▨	🟥	🟥						3		3
Clip handle	009	3														🟥	▨	🟦	🟦					1	2	3
Inner guide	010	6											🟥			🟥	🟦	▨	🟥					3	1	4
Outer guide	013	11															🟦	🟥	▨	🟥	🟥	🟥	🟥	5	1	6
Screws 01	012	15																	🟥	▨				1		1
Screws 02	012	16																	🟥		▨			1		1
Screws 03	-	17																	🟥			▨		1		1
Screws 04	-	18																	🟥				▨	1		1
Geometric pairing			1	2	2	1	5	2	2	1	5	1	2	1	1	3	1	3	1	1	1	1	1	42		
Adjacency			1	1	1	1	1		1		1	2	2	1		2	1	1							16	
Total No Interactions			2	3	3	2	6	2	3	1	6	3	4	2	1	5	2	4	1	1	1	1	1			58

Figure 7.43 – Matrix representation for Door latch (original design) - Banded matrix.

7.6.2 Door latch – suggested redesign

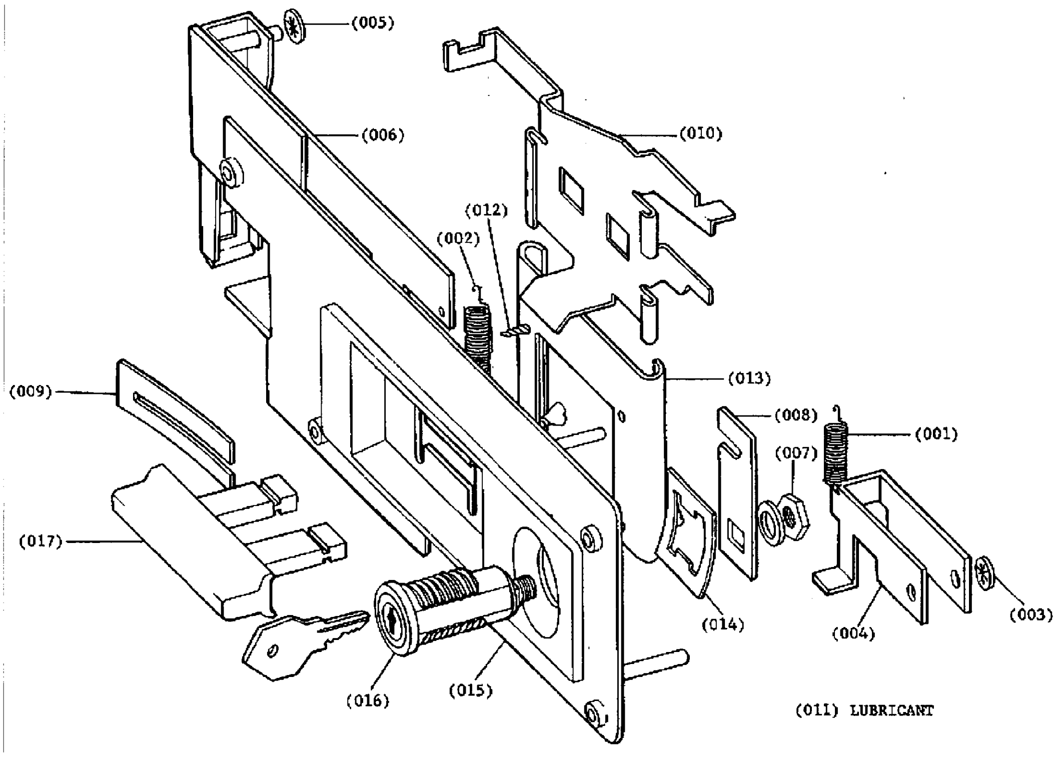


Figure 7.44 – Door latch suggested redesign.

7.6.2.1 Matrix and graph representation

	Components		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17			
001	Spring 01	1			■	■											■					
002	Spring 02	2										■					■					
003	Push clip	3	■			■											■					
004	Bracket release	4	■		■												■					
005	Push clip	5						■														
006	Latch arm	6					■						■	■			■					
007	Nut	7								■								■				
008	Lock arm	8							■			■				■		■				
009	Clip - handle	9										■										
010	Inner guide	10		■						■	■				■				■			
011	Screw 01	11						■							■							
012	Screw 02	12						■							■							
013	Outer guide	13										■	■	■								
014	Circlip	14								■								■				
015	Housing	15	■	■	■	■		■										■	■			
016	Lock	16							■	■						■	■					
017	Handle	17									■						■					
	Geometric pairing		3	1	2	2	1	4	1	1	1	2	2	2	3	1	6	4	2	38		
	Adjacency			1	1	1			1	3	1	2				1	1				12	
	Total No of Links		3	2	3	3	1	4	2	4	2	4	2	2	3	2	7	4	2			50

Figure 7.45 – Matrix Representation for Door Latch (suggested redesign).

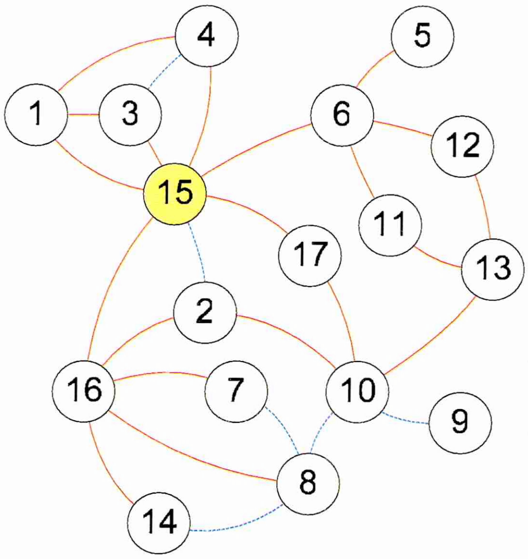


Figure 7.46 – Graph representation for Door latch (suggested redesign)

7.6.2.2 Complexity evaluation

Number of components – n	17
Number of component types – m	14
Number of links – L	50
Connectivity ratio – (L/n)	2.94
Deviation from simple vertical assembly	47.1%
Total cohesion or architectural compactness	18.4%
Component variation factor – (m/n)	0.82
Interfaces	
Geometric interfaces	38
(Amount of geometric interfaces)	76.0%
Adjacent interfaces	12
(Amount of adjacent interfaces)	24.0%
Ratio (Adjacent/Geometric)	31.6%
Interaction saturation	
Total number of actual interactions (LT)	25
Minimum number of interactions (LB)	16
Maximum number of possible interactions (UB)	136
Level of interaction saturation (LS)	7.5%
Unitary-weighted Assembly Complexity Index	
Assembly complexity index without variant factor	1.47
Assembly complexity index with variant factor	1.21

7.6.2.3 Overall interface distribution

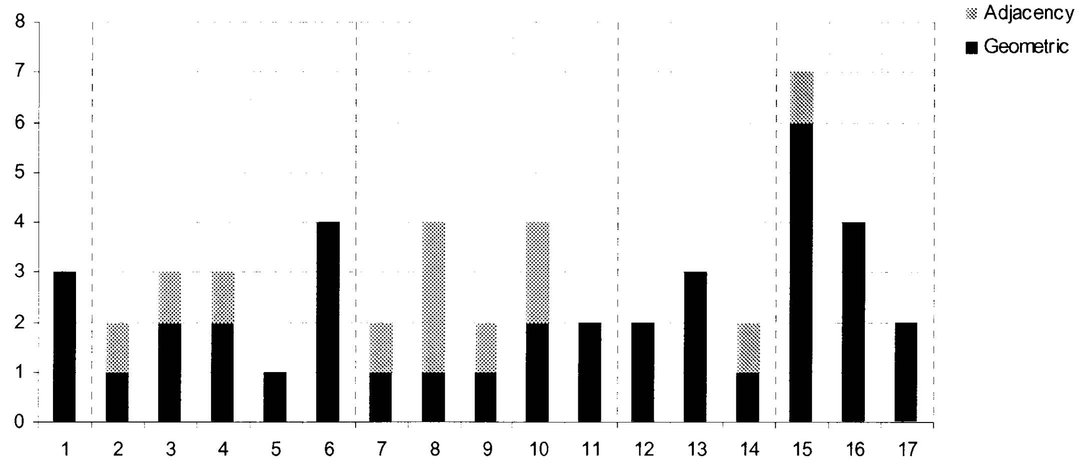


Figure 7.47 – Door latch (suggested design): Overall interface distribution.

7.6.2.4 Comments

This case study in particular presents interesting results, for the final assembly complexity value is higher than that of the original design. This situation is due to the decrease in the number of components. The ACI is inversely proportional to the square of the number of components. Since the "weights" of the type of interactions (interfaces) are not considered yet, then it immediately reflects a value that seems to defy the nature of product design optimisation. This "irregularity" is also due to the increase in the number of links (L). The suggested re-design is more compact and therefore any one component interacts with a larger number of components. Other observations are presented as follows.

- *Connectivity and vertical assemblability* – In this case the connectivity has increased. The deviation from vertical assemblability has been increased as well. Once again, it means that simultaneous insertions manoeuvres are taking place.

- *Cohesion and Interaction Saturation* – The total cohesion has increased slightly, and the interaction saturation has been doubled, but not much improvement in this area.

- *Component variation* – There has been some gain in the component variation factor. At least the number of fastening components has been reduced significantly. Nevertheless, there is a small number of these fastening components that skews the Component variation factor.

- *Interface variation* – Not much of an improvement in this area. Perhaps the increase in geometric interface level is due to the requirements of the design. Parts need to have relative movement and therefore no-mirrored interfaces appear.

7.7 Results summary

The case studies presented in this section do not represent a mathematical proof of the proposed complexity metrics. Evaluations of every case study have been chosen as a verification of the suggested theoretical results presented in chapter 6. Every case study has been selected according to how well they illustrate the benefits of applying product complexity metrics during early stages of the product design.

Some case studies noticeably highlight the improvements of DFA implementation. For instance, case study 05 (wiper motor) illustrates the benefits of component integration, increasing geometric awareness. Case study 01 (pressure relief valve), on the other hand, shows a product structure with zero complicatedness. The latter, in particular, underlines the concept of "minimum complexity levels".

Other case studies suggest results that might initially strike the reader as odd, which is the situation of case study 06. This case (door latch) presents a re-design with a higher assembly complexity value. This higher value is down to two factors: (i) the suggested re-design has a lower number of components, but a higher number of links, than that of the original design, and (ii) no interfaces (or type of interactions) have been assessed at this stage. Case study 06 (door latch) specifically highlights the problems of considering the value of product complexity as solely dependent on the number of components. This case clearly shows that the number of interactions plays a vital role in the ultimate value of the product complexity.

In general, all case studies show that the suggested re-designs tend to favour the increase of "geometric interfaces" over "adjacent interfaces", thus increasing the geometric awareness throughout the product. These, and other specific situations, are summarised and commented on in Table 7.1 as follows:

Table 7.1 - Case studies result summary

State	*Characteristic*	*Comments*
Product: 01 - *Pressure relief valve*		
Original	*Periodicity*	No repeated components. Levels of complexity are not "inflated". *Zero complicatedness.*
	Interfaces	Only one non-geometric interface. Such an interface will be modified later on, in the re-design, from an "adjacent" to a "geometric", thus improving “geometric awareness”.
Re-design	*Connectivity*	Perfect connectivity ratio (connectivity = 2.0).
	Vertical assemblability	*Zero deviation.* This product structure exemplifies the optimum value suggested by DFA methodologies.
	Cohesion	Every component affects two and only two other components directly. This chain-like structure has been selected as the optimum stage, accentuated by a perfect vertical assemblability. (Cohesion = 50%).
	Interfaces	High "geometrically awareness". Every component is “aware” of the presence of its neighbours. Geometric information is transferred successfully. Beneficial for an assembly-oriented design.
	OTHER	Product structure exemplifying *“minimum complexity level”*. Every component fulfils its intended functionality. *Zero complicatedness.*
Product: 02 - *Oil pump*		
Original	*Connectivity*	High level of connectivity, yet lower cohesion levels.
	Localised interaction intricacy	High connectivity is due to localised interaction rather than connections of “one-to-many” components. Attention should be paid to these areas during assembly planning. (localised use of resources)
	Cohesion	Loose architecture suggesting formation of sub-systems (characteristic of complex systems).
	Component characterisation	A non-general characterisation of components fails to describe more accurately the overall complexity of the product (standardisation of component types is required).
	Interfaces	Large number of non-geometric interfaces. Poor transmission of geometrical information throughout the product structure.
Re-design	*Vertical assemblability deviation*	Large value for deviation from vertical assemblability, although product’s vertical assemblability is accepted. Possible cause of inadequate value is due to simultaneous insertion of components.
	Cohesion	Increase in product cohesion (Beneficial due to lower part count and component integration). High cohesion reduces "branching", avoiding formation of sub-systems. (Unless required, sub-systems increase product complexity).
	Component variation	Correct identification of type of components greatly improves the real representation of product complexity. An erroneous classification of components as of different type (i.e. gear and pinion) leads to an inflated level of complexity. Incorrect classification of component types prevents assembly information reutilisation.
	Interfaces	No evident increase in geometric information transfer. Attention should be paid, if an assembly-oriented design is to be followed.
Product: 03 - *Staple remover*		
Original	*Cohesion*	Product structure ramification in visible two branches. Product symmetry, unless required modifications to "one branch" are not reflected on the other. However, product structure is well-balanced, which means that symmetry can be a required characteristic.
	Component variation	A significantly low factor in component variation. Product symmetry implies that assembly information can be extracted with half of the components.

State	*Characteristic*	*Comments*
	Interfaces	Geometric information is transferred throughout the product structure. It does not guarantee that product symmetry can remain due to this transfer.
Re-design	*-n/a-*	No assembly required in re-designed product.
Product: 04 - ***Motor drive***		
Original	*Vertical assemblability deviation*	Vertical deviation due to simultaneous assembly y operations. Fastening components visibly distort vertical assemblability value.
	Localised complexity	Two components dominate the interactivity drawing all assembly and manipulation resources towards them.
	Cohesion	Loose structure - sub-systems formation.
	Component variation	Low component variation - Large number of components are repeated, mainly due to fastening components.
Re-design	*Vertical assemblability deviation*	Vertical assemblability is improved. It still presents a high deviation factor due to simultaneous insertion operations taking place.
	Interaction saturation	Product is more compact. Pair-wise interaction is increased and modifications to one component will affect a larger number of the remaining components. (Combination of components and lower part count).
	Interfaces	High "geometry awareness". Geometric information is assured to be transferred throughout the product structure.
Product: 05 - ***Wiper motor***		
Original	*Component clustering*	Beneficial!! Identification of parallel assembly and modular architecture.
	Vertical assemblability deviation	Deviations due to turnover operations skewed the results of vertical assemblability evaluation.
	Cohesion	Loose architecture - creation of sub-systems and sub-assemblies (is this intentional?)
	Localised complexity	Loose architecture. Areas that concentrate a large amount of assembly resources are easy to identify.
	Component variation	Low component variation - Large number of components are repeated. Arrays and sets of components abound.
	Interfaces	Large number of non-geometric interfaces (geometric information is not transferred throughout product architecture).
Re-design	*Cohesion*	Cohesion levels increased dramatically. More interaction between components indicates that components "share" more information, perhaps fulfilling more specific and essential tasks?
	Component variation	Decreases. Just one part is repeated. Detrimental as no assembly information is re-used, but time is saved in the lower number of components.
	Interfaces	Increase in geometric awareness. Existing non-geometric interfaces are necessary for operational purposes.
Product: 06 - ***Door latch***		
Original	*Connectivity*	No critical information can be obtained. All connectivity levels fall within accepted ranges.
	Cohesion	Low cohesion level. Loose architecture.
	Interaction saturation	Loose architecture. Formation of sub-systems (if not intentional, attention should be paid to this factor).
	Component	Large number of fastening components present a larger than normal product

State	*Characteristic*	*Comments*
	variation	complexity, thus adding complicatedness.
	Interfaces	Large number of non-geometric interfaces (geometric information is not transferred throughout product architecture).
Re-design	*Vertical assemblability deviation*	Deviation from vertical assemblability has increased. This situation is due to simultaneous insertion operations that disrupt this metric.
	Cohesion	Cohesion increases. More compact architecture, Modifications to one component will affect neighbouring components.
	Component variation	The number of fastening components has lowered leaving components of dissimilar types. Once again, this is beneficial as redundancies are lowered and complicatedness is reduced.
	Interfaces	Large number of non-geometric interfaces (necessary for relative movement).
	OTHER	Complexity value has increased, producing a less complicated product, but more complex in comparison with the original. Complexity can also be beneficial if operational requirements demand it. This value is higher, oddly enough, to the decrease in the number of components and the increase in the number of links. The more interactions a product has, the higher the complexity it reveals.

Chapter 8

COMPONENT COMPLEXITY ASSESSMENT

Component complexity is marked with interesting properties. It can be the leading factor in the complexity of the whole product or it can be considered as a by-product of that of the assembly.

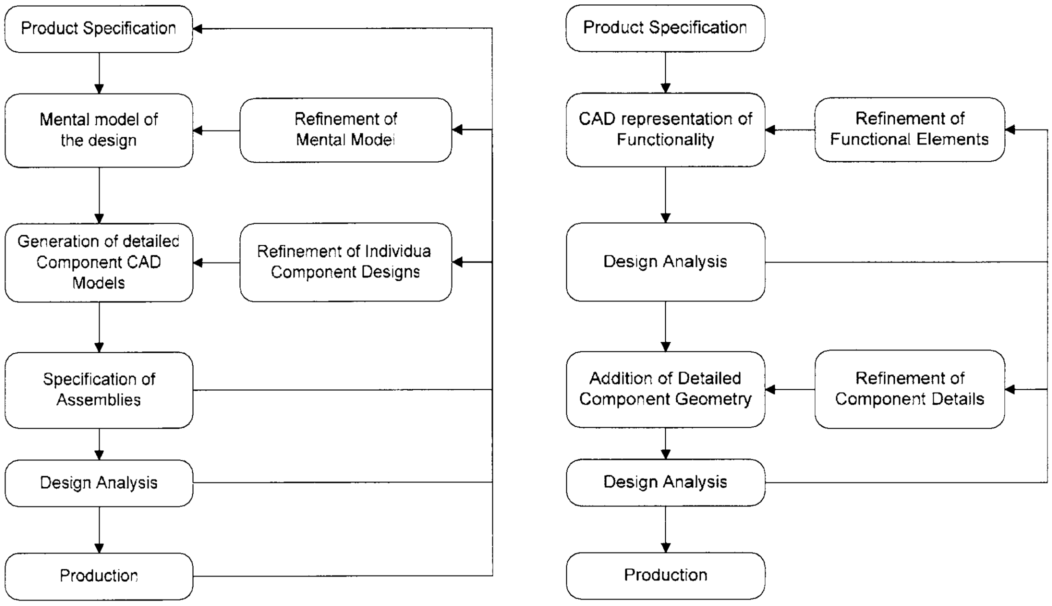
When considered as a leading factor, components are built according to the purposes or tasks they are intended to fulfil, as in a typical design methodology that is. Once all parts[36] have been sketched out, then come the thinking out of their interactions (this accounts for the complexity of the assembly), followed by their assembly order (sequence), which ultimate reveals the complexity of the assembly. This simplistic summary obviously overlooks all the problems highlighted by DFA methodologies, namely, high part count, low design efficiency, poor manufacturability analysis and an extremely inadequate part-to-part interface design.

Most current commercially available CAD packages still tend to concentrate upon 'component-oriented' design, where individual parts are modelled and then assembled to create the final product. This is not an effective way to ensure that designs are translated to efficient manufacture and assembly and can lead to problematic assembly, rework and ultimately redesign. Hence, a need has been identified for development of an 'assembly-oriented' CAD environment to support a more 'top-down' approach to the design process

The findings of Sodhi and Turner [169], who propose an assembly-modelling environment as the key to representation and maintenance of functional intent in product design, reinforce this view. They note that the 'bottom-up' CAD approach is

[36] Parts, which in combination, are supposed to accomplish the tasks the product was designed for.

not consistent with, and therefore does not adequately support, the design process. In a comprehensive review of the literature they discuss the representation of function in terms of assembly features and component interactions. Several investigators have pursued the development of appropriate representational models that will then permit subsequent reasoning for product structure definition and assembly sequence generation. Integration of these facilities has been proposed to enable assembly-oriented CAD and top-down design support within a useful and appropriate software environment.


Product Specification
Mental model of the design
Refinement of Mental Model
Generation of detailed Component CAD Models
Refinement of Individual Component Designs
Specification of Assemblies
Design Analysis
Production
Product Specification
CAD representation of Functionality
Refinement of Functional Elements
Design Analysis
Addition of Detailed Component Geometry
Refinement of Component Details
Design Analysis
Production

(a) (b)

Figure 8.1 – Design Methodologies; (a) Bottom - Up vs (b) Top – Down design

(Adapted from Sodhi and Turner [169])

On the other hand, the second option is represented by an assembly-oriented design environment, such as that of the Designer's Sandpit project documented in chapter 2. Components tend to behave as pieces that fit in the slots outlined by the assembly structure layout. Interactions and those interfaces involved (see Chapter 5 for a definition of 'interfaces') suggest that the complexity of components ought to be seen as a by-product. Consequently, the geometric shape of a component is heavily influenced by the interactions it has with other components. Interfaces mould out most of the component shape. This shape is then a collection (set) of all interfaces found in the component interactions (with other parts that is) plus the specific shape required to perform the task the component was needed for in the product in the first place.

As shown in Figure 8.2 (a), components are constantly "reminded" that they belong to a greater structure, therefore their interactions, and interfaces, are constantly checked in an assembly-oriented design environment. Figure 8.2(b) (mobile telephone case) could be used as an example, where its overall shape has been designed to work as the main body or framework of the whole product, but it has also been designed to accommodate other components that need be integrated onto it.

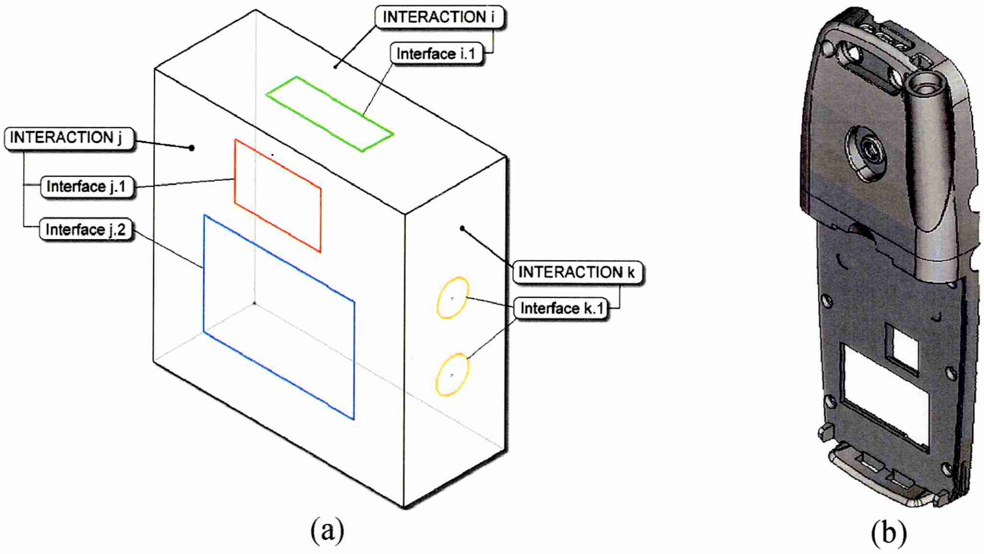


Figure 8.2 – Component Shape; (a) A collection of interfaces that determine the component's overall external shape; (b) Component shape example

However, there are components that are completely "unaware" of their neighbours, they do not "know" that they belong to a greater product structure. The reason for this is the differences in component-to-component interface types (geometric and adjacent interface types, as seen in chapter 5). Geometric interfaces "carry the information" of the assembly to the component, hence when the component is manufactured, these interface's geometric data is embedded in the component's shape. Interfaces that are primarily adjacent "do not carry" this information. Components that exclusively have adjacent-type interfaces are bound to introduce complications in the assembly process. It is as if the interaction between this sort of component and its neighbour happened by accident.

8.1 Component complexity Classification

As previously described in chapter 5, product complexity can be sub-divided into that of the assembly and the set of those factors that affect every component, comprising the component's complexity. Component complexity can be further sub-divided into manufacturing and manipulation related complexity (see Figure 8.3). This is the subject of the remainder of this chapter.

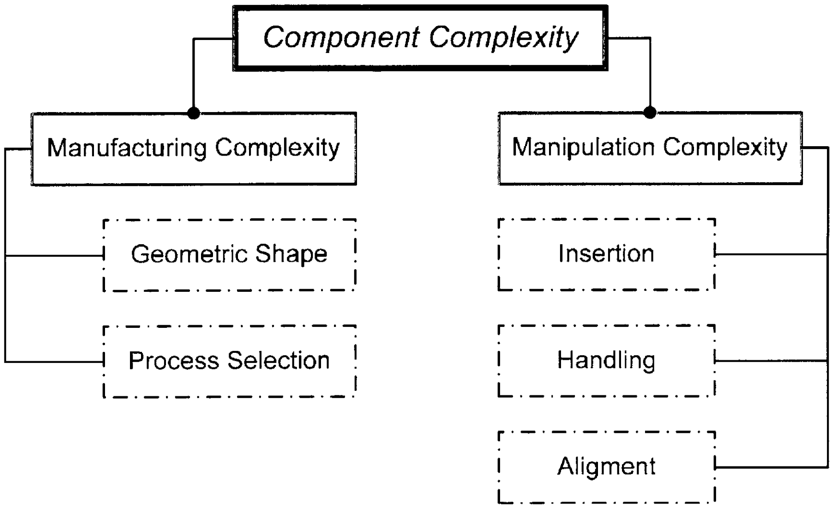


Figure 8.3 – Component complexity classification

8.2 Manufacturing complexity

Product development involves several activities that often use similar terms, but refer to different actions. For instance, the expression "component manufacturing" regularly suggests activities such as fabrication (geometry), assembly, fastening, inspection and so forth. In order to avoid confusing misinterpretation, terms such as "manufacturing", "manufacturability" and "fabrication" as addressed in this chapter, refer exclusively to the achievement of the component or artefacts geometric shape. Consequently, the evaluation of the component geometric data plays a vital role in the estimation of its manufacturability.

8.2.1 Geometric information and shape complexity

The classification of shape is important in many aspects of engineering. It is often useful to group parts by their required manufacturing processes or common features, as addressed by group technology codes. In addition, the recognition of features and other aspects of shape are useful for the automatic generation of tooling paths. However, shape complexity is also a major factor in the determination of component manufacturing difficulty and associated costs. Therefore, the Manufacturing Analysis section of the Lucas DFA (and its derivatives) uses a measure of shape complexity as the basis for all analyses. In this context, shape complexity is used to quantify the manufacturing complexity of objects, to calculate relative estimates of associated costs and to highlight incompatible processes and potential problems. For instance, complex surfaces such as b-splines will almost always require dedicated tooling and thus costs are increased. Another design dependent characteristic that affects manufacturing costs is the existence of multiple axes in a component, which thus requires many component re-orientations during manufacture.

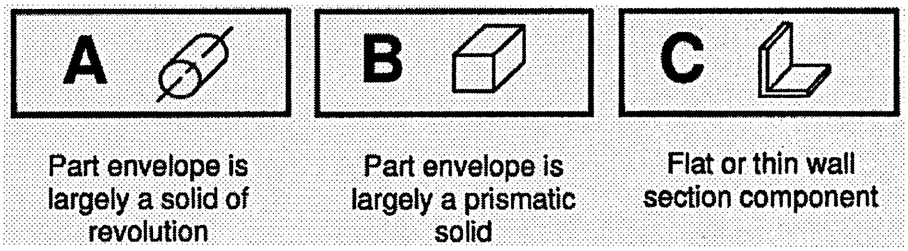

A
B
C
Part envelope is largely a solid of revolution
Part envelope is largely a prismatic solid
Flat or thin wall section component

Figure 8.4 – Component classification based on overall shape

In particular, components (Figure 8.4) are divided into three main types:

- *A parts* – the part envelope is largely a solid of revolution;
- *B parts* – the part envelope is largely a rectangular or cubic prism;
- *C parts* – flat or thin wall section components.

Within these categories, components are then ranked on a scale of 1 to 5 according to the number and types of features and the complexity of the surfaces. (See Appendix III for information about traditional shape classifications). These 'definitions' of shape complexity illustrate the vague and subjective nature of shape classification in general and thus they highlight the difficulties of automating its DFA procedures. In order to create a useful computer-based tool for DFA analyses, the definitions of shape complexity must therefore be adapted to create a measurable and objective system of classification.

There have been several attempts to classify objects in engineering environments, mainly for the purposes of automating their manufacture or grouping similar parts. Automated methods have tended to rely upon feature recognition approaches to the problem. Shah [170] provides a useful introduction to the techniques devised in this area. However, many obstacles exist in defining globally successful algorithms. Problems due to feature interactions [171] and the industrial interpretation and language used to describe particular features, are typical.

In DFA, shape complexity is used to determine manufacturing difficulty and so techniques aimed at automating manufacture are worthy of further examination. There are several techniques in this area. The volume decomposition technique of Woo [172] identifies the material to be removed from the base stock and breaks it down into units corresponding to machining operations. This method is typical of many such algorithms that seek to identify the shapes of material to be removed. An alternative method, based on the boundary representation of CAD models uses graph connectivity to generate tool paths without actually recognising features [170]. However, neither of these techniques is appropriate for the purposes of a DFA Manufacturing Analysis. The requirements of an algorithm for determination of shape complexity do not include the explicit recognition of features, per se, but the quantification of those features. Neither is the number and complexity of particular *machining* features, sufficient to quantify shape complexity. The algorithm must identify the basic types of shape (A, B, and C) already defined by the DFA methodology, since the shape complexity index must be related to many types of manufacturing process, not just machining.

An alternative approach to the shape complexity problem lies in methods for group technology part coding. Many of the methods for shape classification are based on the ideas of Kyprianou [63] whereby the boundary representation graph is examined to identify convex and concave relationships at vertices and edges. A few years later, Ames [64] combined these techniques with an expert system approach to the automatic generation of part technology codes for rotational, prismatic and sheet metal parts. An alternative method is presented by Little *et al* [70] which was primarily defined to quantify complexity for the features technology research community. The method does, however, relate the shape of a component to its manufacturing complexity by the number and orientation of different face types. A string is formed based on the types of face, the number of 'aspects' determined by evaluating the surface normal of all planar faces, and the number of cycles of faces that can be generated.

The latter methods partly fulfil the requirements of an algorithm for evaluation of shape complexity in the context of DFA. However, an alternative interpretation is required in order to relate the shape complexity to particular manufacturing processes as required by the DFA methodology.

8.2.2 Shape complexity – definitions

The starting point for the redefinition of shape complexity is an appraisal of the fundamental purpose of the Manufacturing Analysis. The written definitions and terminology used to describe shape complexity offer more insight into this purpose than the shape complexity categories themselves. These definitions, extracted from the Lucas DFA methodology [11] are as follows:

- *Basic Features* – Straight forward processing where the operation can be carried out without a change of setting or complex tooling. Usually parts are uniform in cross-section.
- *Secondary Features* – As above but where additional processing or more complex tooling is necessary.
- *Multi-Axis Features* – Parts require to be processed in more than a single axis/set-up.
- *Non-Uniform Features* – Parts require the development of more complex processing techniques/set-up.
- *Complex Forms* – Parts need dedicated tooling and the development of specialised processing techniques.
- *Single Axis* – This is usually the axis along the components largest dimension. However, in the case of cylindrical or disc-shaped components, it is more convenient to consider the axis of revolution as the primary axis.
- *Through Features* – Features which run along, across or through from one end or side to the other.

Although still somewhat vague, these descriptions highlight the fundamental principles of the Manufacturing Analysis, which relates manufacturing difficulty to the number and complexity of the manufacturing processes required. The Lucas

DFA Methodology incorporates tables that show how the difficulty of particular manufacturing processes is related to the shape complexity of a component. Several different manufacturing processes may be able to deal with the features defined and thus process capabilities and costs strongly influence the choices made. For instance, a specific tolerance or surface finish may be achieved by the primary manufacturing process, but a cheaper manufacturing process combined with an appropriate finishing technique may be more cost-effective. The difficulty of producing a component using a particular manufacturing process is also dependent on such factors as the component material and the minimum section thickness. All these factors are considered and combined in the calculation of a 'Manufacturing Index' to enable a simple comparison of alternative designs.

In summary, the difficulties and costs associated with component manufacture are directly related to the number of types of manufacturing process required, the number of faces requiring each manufacturing treatment and the number of tool or orientation changes required during each phase of the manufacture. The following rules can be inferred, which correspond to this premise:

- Particular manufacturing processes can be associated with particular types of surface. For instance, cylindrical faces tend to be created by either turning or drilling operations, which are simple processes, whilst curved surfaces tend to be produced by casting for solid objects or by some type of stretch forming or extrusion operation for thin-wall sections. The production of b-spline surfaces is generally more difficult than rotational or planar surfaces, since dedicated, and therefore costly, tooling is generally required.

  ➢ *Shape complexity is related to the number of types of face*

- The number of component re-orientations is dependent upon the number of faces that can be produced by a single component/tool configuration and hence is related to the number of co-oriented or coaxial faces.

  ➢ *Shape complexity is related to the number of different face orientations*

- The number of manufacturing operations required is related to the number of so-called 'features' and their level[37] with respect to the global shape. For instance, a geometric feature would not generally be produced until its parent face has been produced.

  ➢ *Shape complexity is related to the number of levels in the face hierarchy*

- Material removal for creation of depressions is less complex than material removal for creation of protrusions, in many cases. The obvious exception is the creation of a narrowed end section in a turned component (Figure 8.5), but in this case the face is parallel with the primary axis.

  ➢ *Shape complexity is related to the number of protrusions not parallel to the primary axis.*

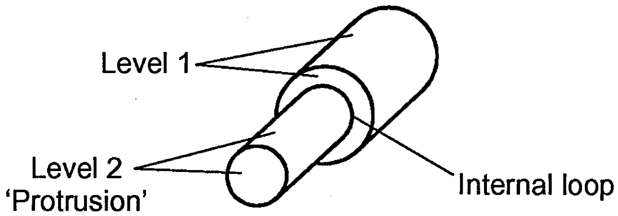


Figure 8.5 – Axial 'Protrusions' in Rotational Components

8.2.2.1 Shape Complexity Quantification – Initial approach

Shape complexity can be determined for each component, but for a better understanding of their combined effect in an assembly, this must be calculated such that it may:

- Be used in conjunction with other metrics,
- Be combined in calculation of configuration complexity,
- Enable comparison of shape similarity (for reduction of variance).

One initial approach was originally exposed elsewhere [173]. This method aimed at ranking shape complexity in terms of a shape-based hierarchical tree, where measurements could include the number of levels (depth) or branches (breadth). Consider such a tree structure in relation to component topology: The top level (root) is the CAD model as originally presented. Subsequent levels represent the

[37] A face hierarchy can be constructed based on whether particular faces are externally bounded by the internal loop of a 'parent' face, see Figure 8.6.

object obtained by iteratively stripping the model of its features. When the simplest shape (basic primitive) has been achieved, then the numbers of levels traversed will be one measure of component complexity. The number of features identified at each stage will define the breadth of the tree and will be another measure of the component complexity. The primitive and features obtained at each stage could also be used to identify similar objects. Incidentally, this practice required a mix of Boundary Representation (B-rep) and Constructive Solid Geometry (CSG) representations.

Inevitably, the problem is not so straightforward: some types of surface are more complex than others and require more difficult manufacturing processes. For instance, b-spline surfaces may be considered more 'complex' than planar ones since they often require dedicated tooling. Therefore some extra '*weights*' must be added to the complexity measurement to account for this. Defining the precise geometric reasoning methodology and additional factors will form the basis of the research.

8.2.2.2 Loop-based quantification of shapes

The above described procedure attempted at exploiting early approaches to shape evaluation. One such method required the calculation of loop properties in the B-rep model.

This step involves the determination of surface type, loop type (internal or external), loop area, loop centroid, surface normal at the loop centroid and the number of edges in the loop, for every loop of every face. The loop properties are then written to a database for storage and subsequent manipulation. These properties are defined in more detail elsewhere [75, 174]. (This method was also extended to calculate rotational symmetry in solid models [175])

The method proposed here is typical of many feature recognition algorithms, which locate features by the existence of internal loops [176], a technique originally devised by Kyprianou [63]. In this application, the recognition of 'features' per se is not required but internal loops provide a starting point for creation of a hierarchy of faces. Level 1 faces are those for which the external boundary loops do not contribute edges to the internal loops of any other faces. Level 2 faces are those with one or more edges contributing to an internal loop of a Level 1 face. Level 3 faces are those with one or more edges contributing to an internal loop of a level 2 face, and so on. The face level is determined as follows:

Using the properties calculated in Step 1, the internal loops, with their associated faces, can be extracted from the database and ranked according to the bounded surface area. The algorithm then performs a hierarchical search to find the 'child' faces of all faces.

Taking the face with the largest internal loop as the starting point, a check is performed to ensure that the external loops of the face do not contribute edges to any other internal loops. This guarantees that the starting face is a Level 1 face. By starting with the largest internal loop the likelihood of selecting a lower level face is minimised but should this occur then the algorithm 'steps up' a face level and repeats the check. The internal loop of the Level 1 face can then be used as the origin for searching this particular portion of the hierarchy. For each edge of the internal loop, the solid model can be used to find the corresponding external loop of the next level face as shown in Figure 8.6. The internal loops of this face are then treated similarly and the process continues until the lowest level of face in the chain has been identified. As each stage of the search is completed, the faces that have been addressed are deleted from the possible search space so that repetitions and contradictions do not occur. The faces remaining when all internal loop threads have been followed are, by deduction, Level 1 faces. By grouping same-level faces in each thread of the hierarchy, a very basic 'features' list can be created.

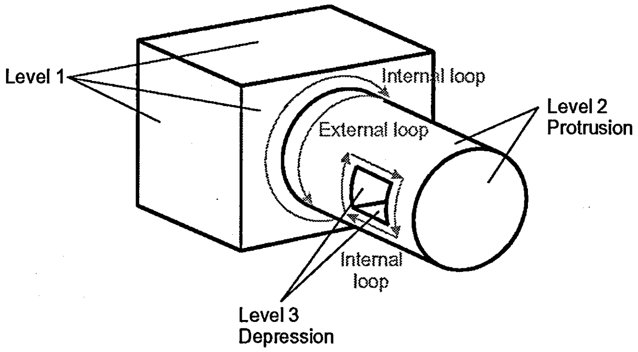


Figure 8.6 – Parent-Child Face Relationships

Following the extraction of loop properties, a face hierarchy was constructed, as well as examining the relationship between parent and child faces. Only one edge of each loop is required to perform a check to determine whether the child faces form a protrusion or a depression in the parent face (see Figure 8.6). This technique is also attributable to Kyprianou [63], and is commonly used in feature recognition algorithms. When this information is added to the list of basic features, it forms the basis for further classification of the features. However, for the purposes of this algorithm further analysis is not required.

Further information to be extracted during this procedure implied the determination of global shape, that is, the classification of components according to the A, B or C overall shape (see Figure 8.4). Finally, evaluations of the components shapes were based on the number of loop-levels and a degree, between 1 and 5, within the particular category, was selected.

Nonetheless, after revising the whole procedure of shape classification and manufacturing process selection, it was found that the method described above could be replaced by another that not only examined the object's shape, but that would ultimately related its shape to a suitable manufacturing process. This could be achieved by taking the process characteristics (i.e. convenience to deal with cylindrical shapes or thin-walled components) to interrogate the component shape. This procedure will be explained in the following sections.

However, the examination of the former procedure is not wasted effort. As previously mentioned in chapter 6, some of the classifications of part-to-part interface require an "interface evaluation index/value". The best approach would be to find such an 'interface value' through manufacturing analysis, and therefore associate it with its manufacturability. However, this value can also be obtained using the loop-based algorithm already suggested. The loop-based algorithm, as it is proposed, does not associate the interface value with its manufacturability, but it would help to create a particular scale of geometric intricacy. This scale would be useful to rank every interface accordingly and to enable the comparison of any two interfaces.

There are no case studies to support a geometric interface ranking system based on a loop-based algorithm, therefore it is proposed as part of the further work that could follow from this thesis, further complementing the complexity metrics proposed.

8.2.2.3 Shape complexity and manufacturing process selection

The shape complexity evaluation procedure suggested by many DFM methodologies suggested branding the component's overall shape as either A, B or C (part envelope is large a solid of revolution, a prismatic solid or flat/thin sheet metal section, respectively).

This procedure is well suited for human interpretation, as it requires a human operator to examine the component and compare its shape to a set of previously selected components organised in a chart. This chart would suggest the most approximate classification label. Additionally, the operator would have access to a chart of formerly analysed manufacturing methods, which have already been examined in terms of the manufacturability of a particular component. With these two charts –shape and process classification, respectively- the operator would cross link the shape to the process and extract a "shape complexity index." (see Appendix III).

From the above, it can be read that any attempt to classify the shape of a component, in terms of the A, B or C chart, would simply follow an old method of human interpretation. Moreover, the manufacturing processes listed in the "process selection chart" are very limited, if an additional method were to be introduced, it would take a further re-evaluation of the shape classification chart. This limitation demanded a novel approach to shape evaluation for manufacturability analysis, and this approach is described in the following section about "manufacturing process characterisation."

8.2.3 Manufacturing process characterisation

Traditionally [177-183], components have been classified and sorted based on their overall shape and then grouped accordingly in part families and/or clusters with similar characteristics that will respond in a more or less predictable manner to the manufacturing processes available.

However, with the advent of readily available computer power, manufacturability analysis can be taken a step forward. Instead of relying on human-oriented part coding systems, manufacturing processes can be characterised in order to extract their advantages and disadvantages when consulted for their suitability to manufacture a certain component.

In order to explain the proposed method of manufacturing analysis, consider the following scenario. From a whole range of manufacturing processes available nowadays (see Figure 8.7 for a quick summary), Company X has just a certain number of those manufacturing processes at their disposition. Using the old system of part coding, the company has found that most of the components that need be manufacture respond favourably, in terms of shape complexity, to the processes listed in the evaluation chart. However, there are certain processes available to the company (within their budget) that are not listed in the charts, moreover, there are certain components that rank well in processes that are not available to the company.

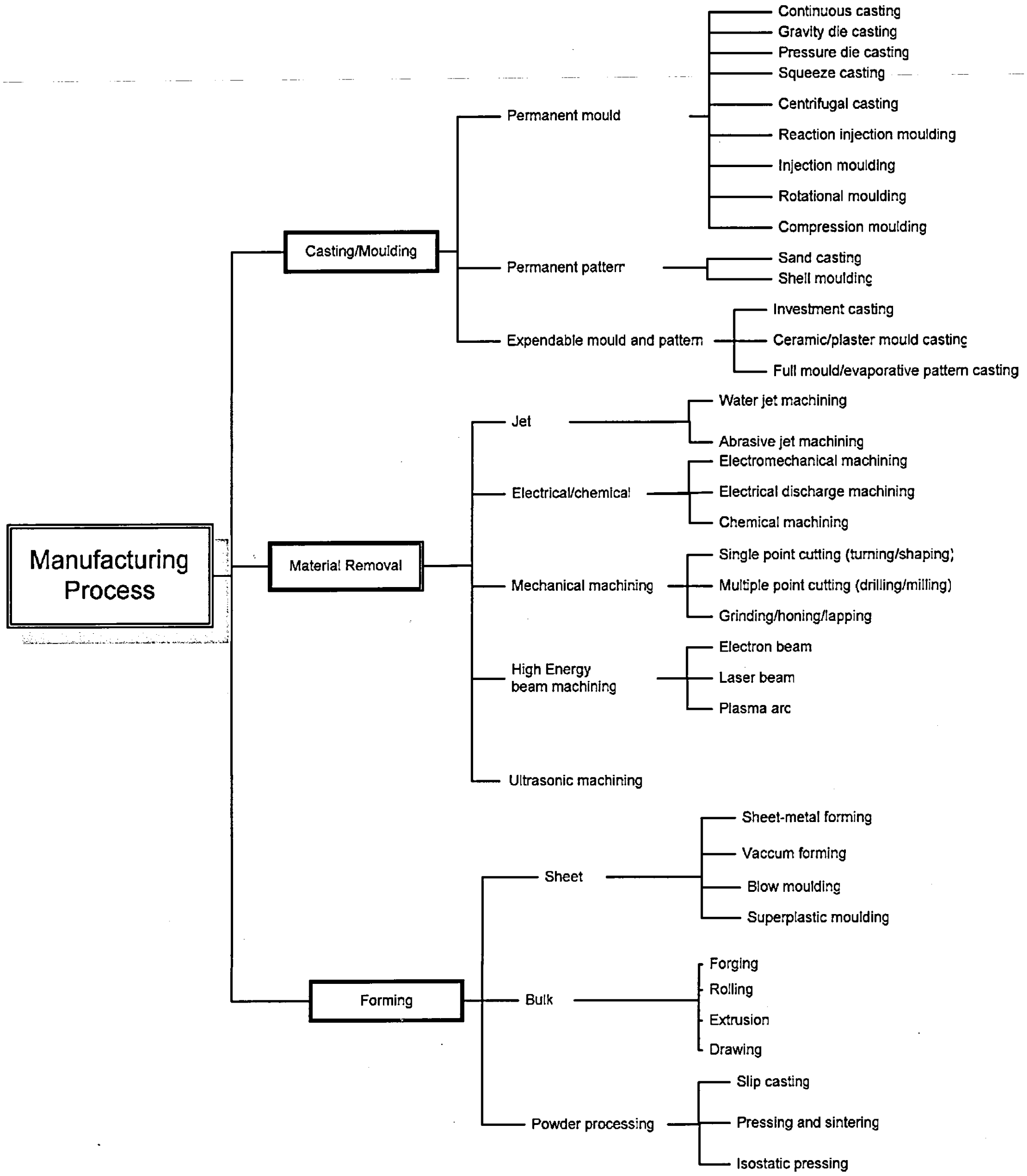


Figure 8.7 – General classification of manufacturing processes *(Adapted from Swift and Booker* **[184])**

The above scenario presents two options. First, the company can obtained an updated version of the manufacturing analysis charts and have the components ranked by the newly acquired charts, that is, if the new charts have more processes listed. Second, the company could produce a list of all the characteristics (pros/cons, advantages/disadvantages, costs, operational costs, amongst others) and with these interrogate the models to evaluate their suitability.

The second option is, without a doubt, a daunting task, if performed by a human operator. Nonetheless, as previously highlighted, computers are quite able to go through a list of characteristics and compare every process's list with the component's own list of characteristics.

This scenario could be branded as "manufacturability on demand analysis." Designers would only consider processes that are available to their industry or organisation. They could, nonetheless, emulate the manufacturability analysis, had other processes been available.

It satisfies the requirements suggested by several authors (e.g. Whitney [185]) when they gave emphasis to design strategies to improve the efficiency of tool management. For instance, Whitney suggested that a "defined tool method" forces product designers to assume the availability of a limited set of cutting tools and must design each part so that this set will be sufficient to make it.

8.2.3.1 Suggested analysis by characterised manufacturing processes

As stated previously, the new approach to component manufacturability analysis is to take the manufacturing process to interrogate the solid model and agree on how suitable (or unsuitable) a manufacturing process is to actually fabricate the component.

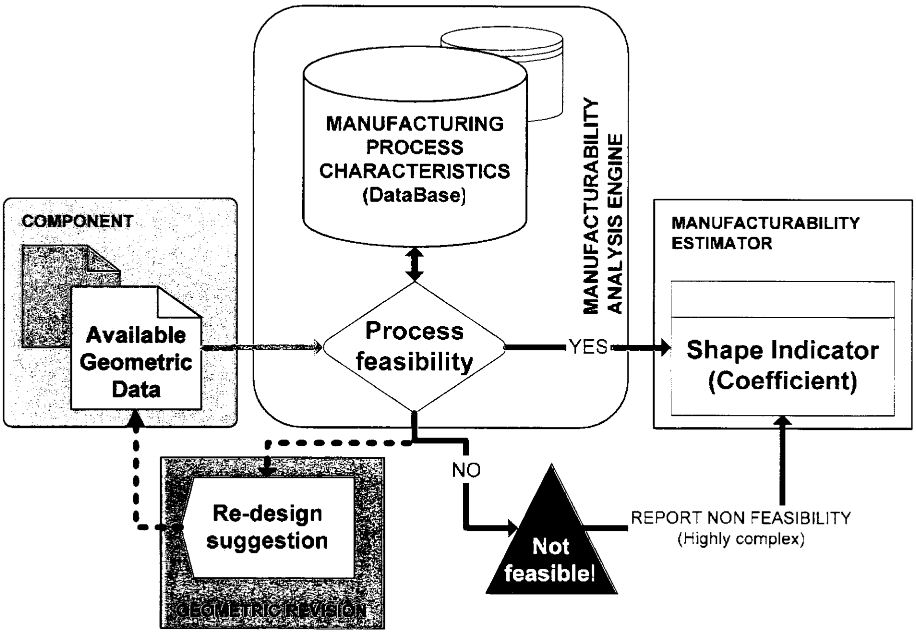


Figure 8.8 – General view of manufacturing analysis supported by process characterisation

The exploration of the component's manufacturability, as depicted on Figure 8.8, requires a database of "manufacturing process characteristics."

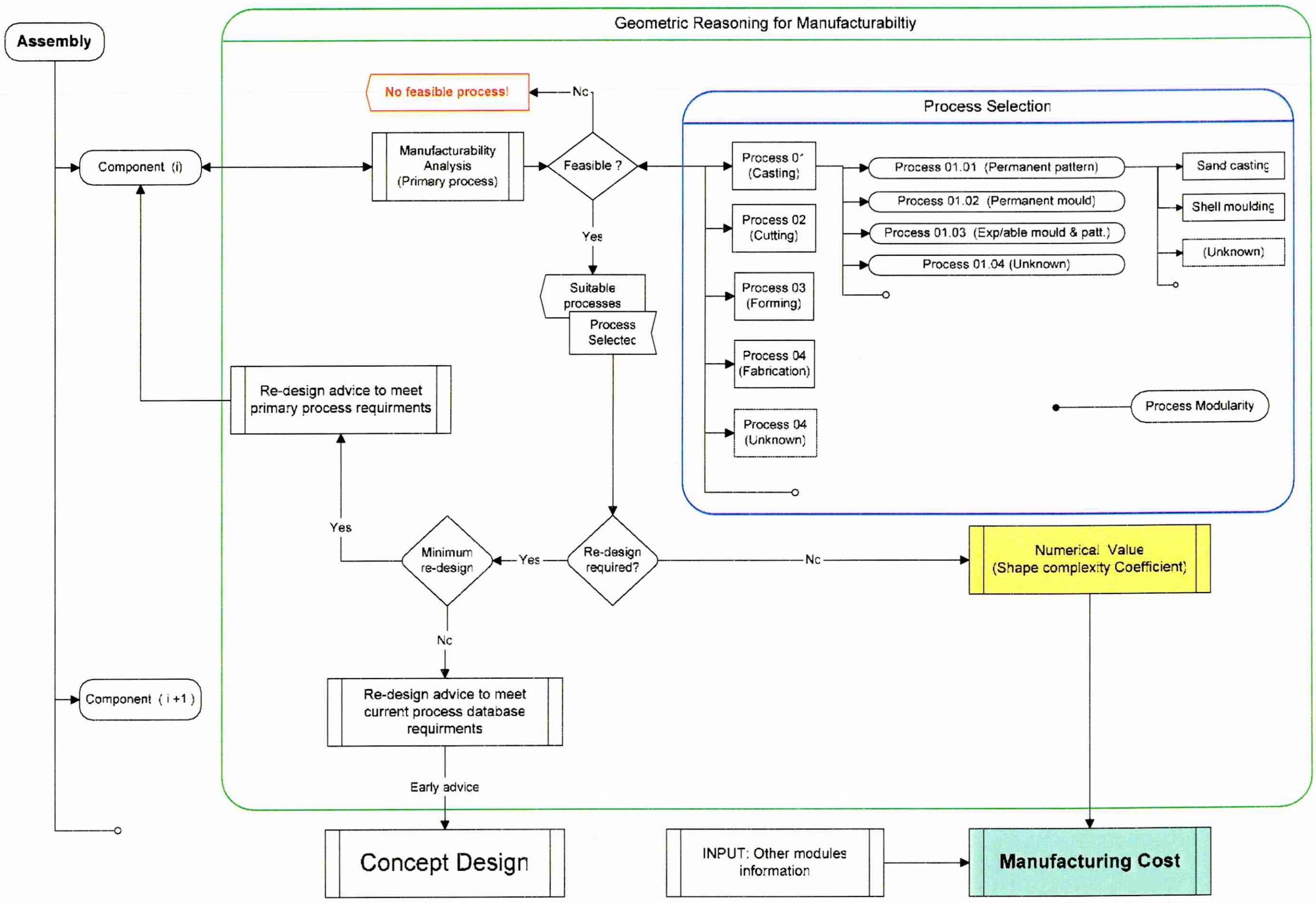


Figure 8.9– Manufacturability analysis supported by a process characterisation engine (detail)

Some of the steps involved in the manufacturability analysis are explained as follows:

Step 1. Primary overall process estimation. Primary process estimation is based on the comparison of the characteristics of the model and those of manufacturing processes stored in the manufacturability analysis engine (database). The general classification portrayed in Figure 8.7 describes a hierarchical decomposition of those manufacturing systems. This organisation is, by and large, based on processes characteristics. Some casting processes, for instance, have limitations regarding wall thickness, sharp corners, and cored holes, amongst others. These characteristics are somewhat common to many casting process and can be used as the initial screening process for primary process selection. As might be expected this initial step benefits dramatically from an early material selection of the component.

Consequence 1. Components characteristics extraction. This activity refers to the interrogation of the geometric data within the model (Figure 8.10). Information such as holes and corners radii, wall thickness (distance and uniformity), maximum and minimum section measurements, area-to-thickness ratios, draft angles, inserts and so forth. The necessary information depends on that extracted from the manufacturing processes, since the processes are the ones limiting the feasibility of the fabrication.

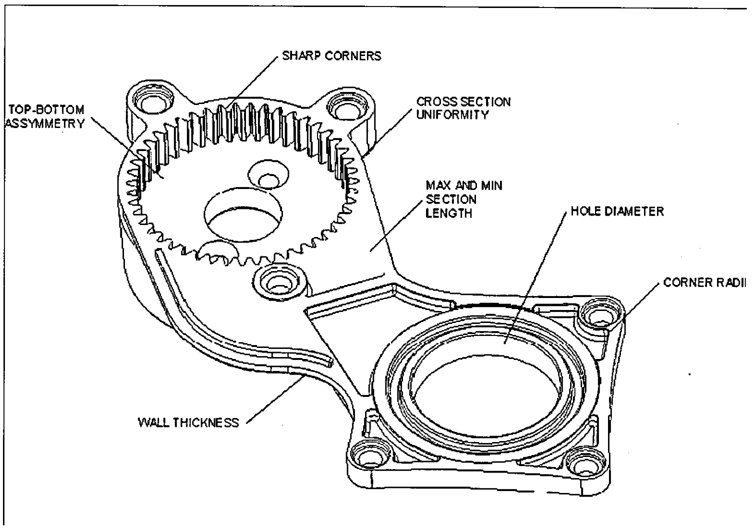


Figure 8.10 – Component Feature characterisation

Consequence 2. No feasible process available. When most of the features found in the model do not meet, under certain tolerance range, those characteristics listed in the predefined manufacturing processes, the model can be initially flagged as "not feasible for manufacture with current processes." This will either prompt a major re-design of the model, in order to meet current process requirements or it will provoke the rethinking of the component shape as it stands.

Step 2. Overall process sub-selection. A process sub-selection is the estimation of the fitness of all the manufacturing procedures listed under a general fabrication technique. If the model has successfully passed the initial characteristic manufacturing process screening, then the major fabrication techniques (general categories, i.e. casting, forming and others) that ranked the highest are selected for subsequent interrogation of their specialised methodologies. For example, if the model has been found to meet most of the requirements of the casting processes, then it is once again examined to establish with which casting process (permanent pattern, permanent mould or expandable mould/pattern) it conforms the most. The procedure continues until all processes classified under the casting process groups have been explored.

Step 3. Ranking of suitable processes. This is the listing of the processes that are fittest to manufacture the component. It is not likely that all processes within a given category respond favourably to the component shape characteristics. The ranking of the most suitable processes will be made only with those that, within a certain range, are fully accommodating of the component's actual shape. The ranking of manufacturing process suitability can be stated as a percentage, which could ultimately be transformed into a numerical coefficient (see Step 5)

Step 4. Selection of most suitable manufacturing processes. The selection of the most suitable manufacturing process from the list lies within the requirements of the user. It can be found that a large number of processes actually match the characteristics set by the model's shape. However, the user must have the option of further filtering the list by either selecting the must convenient processes him/herself or by taking into account additional requirements, such as component material (if not properly selected in step 1, or for material variations, i.e. different types of metal) and quantity of components to be manufacture, to name a few.

Step 5. Slight re-design advice to improve manufacturability. Re-design might be required (or not) for some components to fully comply with the characteristics exposed by the fittest manufacturing process to take on its fabrication. In case the component shape meets most of the process' requirements, then a numerical value can be extracted, based on how closely these constraints are met. Conversely, the mismatch of component shape features and available manufacturing processes would prompt a redesign of the component to be able to meet the fabrication requirements.

Consequence 1. The shape complexity coefficient (numerical value) must be at least equal to one (Coefficient = 1.0). This value is based on the manufacturing cost formula used in DFM analysis[38], for it is a multiplying factor. The list of most suitable process suitability, ranked as by percentage, can be converted into coefficient values for the "shape complexity value", thus a process fit 100% will be equivalent to a Shape coefficient = 1.0, a process fit only in a 10% will be equivalent to a shape coefficient = 10, with the subsequent divisions in between to comprise the rest of the values required.

Consequence 2. A mismatch between the characteristics shown by the geometric shape of the component and those of the manufacturing process will prompt either a slight modification or an overall revision of the geometry of the component.

a. *Slight modification* – From the list of suitable manufacturing processes, it could be seen what causes the fabrication technique to rank poorly. Although the process was rendered as feasible, its poor ranking might have been due to small discrepancies, for instance, a specific casting process might be particularly weak due to the corner radii of the model or the diameter of the cored holes. Such modifications can be highlighted, and if possible, revised to fully comply with the features of the fabrication process.

b. *Overall revision* – The list of suitable manufacturing processes can also highlight certain processes that, although, rated feasible, are ranked extremely poor (e.g. less than 70%). Such an inadequate ranking percentage can suggest that the available processes are not capable of manufacturing the component efficiently. Consequently, the manufacturability of the component might benefit from a general revision of its geometry, given that the modification does not negatively affect the component's performance (what the component was designed for, in the first place) or its interaction with other components.

[38] See Appendix B for information about the manufacturability analysis used in DFM, as suggested by the Lucas DFAM method.

Step 6. Report shape complexity coefficient. Even if a list of manufacturing processes has been drawn up, the characteristics of the component's 'intended' shape might not convincingly match those of manufacturing processes listed. In this case, a warning for the revision of the component might be issued to make modifications to the component shape or overall product structure that has dictated its shape.

The database set to contain the manufacturing processes can be structured in a modular fashion. This will give the chance of actually evaluating the shape of the component to meet only those fabrication techniques accessible to the organisation. Once new processes become available, these can be either activated for evaluation (if already characterised) or incorporated to the database by a new process characterisation.

This approach also offers the benefit of continually exploring manufacturing processes that might be available to the organisation but unknown to the designer/user.

8.2.3.2 Foundations for a manufacturing process characterisation

Since the early 80's and throughout the 90's, academics and practitioners became ever more fascinated by feature technology [186, 187], with Kyprianou as one of its pioneers [63]. In manufacturability analysis, the academic community has embraced even more the idea of feature recognition, either by creating a library of recognisable features [188, 189] or simply by focusing on specific features exhibited by artefacts fabricated with particular manufacturing processes. Manufacturability analysis of domain-specific processes ranging from automatic recognition of machined surfaces [190], graph-based systems for machined-feature recognition [191], die-manufacturing systems [192], manufacturability analysis of machined parts [193-195], sheet-metal parts [196], cast-then-machined parts [197], powder metallurgy [198] to mill-turning process planning [199]. The quest however has bewildered academics with the way in which these features should be recognised and dealt with. Several papers have been published promoting the recognition of one or all of the general shape classification groups mentioned before (i.e. A – solid of revolution, B – prismatic or C – thin walled). The publications vary from automatic shape-classification [182, 183] to automatic recognition of design and machining features [200]. This sprawl of options has led to researchers to question which the features are those that, indeed, need be recognised [201, 202].

Automatic feature recognition processes for manufacturability analysis and design for manufacture have been pushed forward by the ever demanding requirements of concurrent design to integrate evaluation systems early on in the design cycle. Swift [203] explored this type of integration with the creation of knowledge-based systems for design, as did Brissaud and Tichkiewich [204] in more recent years. Gupta *et al.* [205] also suggested the integration of critiquing systems into CAD systems that would eventually empower the manufacturability analysis. This trend was quickly followed by other academics that saw the benefits of integrating AI and machine learning techniques to manage complexities manufacturing [46, 206, 207].

Most of the works reviewed so far have been based on one single premise: "Analyse the artefact, decompose it and look for a specific manufacturing process that can fabricate it", in other words, it meant "taking the piece to the processes" Although, often a single process does not fabricate the whole object, there is a set of features that frequently dictate what the primary manufacturing process might be [188].

There have also been attempts to recognise neutral features that belong to multiple domains [208]. Neutral feature recognition is based on the difficulties found in recognising interacting features. Feature interaction destroys part or all of the characteristics of the basic features. Moreover, this neutral feature recognition moves towards feature recognition independent of a manufacturing process. This approach could expose the model to comparison with several domain-specific manufacturing processes. Other studies are based on multiple interpretations across domains [209] and the associativity between feature models across domains [210]. These works, then again, have opened the door to new explorations in manufacturability analysis. This new trend can be seen later on in this chapter, where manufacturing processes characteristics have been selected as the key aspect to search through a database of processes that can be used to manufacture a particular component.

Based on the old concept of "substance characterisation" used by chemists, whereby a substance undergoes a series of test to establish its nature, artefacts could undergo a series of interrogation procedures about their its geometric shape. These procedures could establish which process is best suited for the fabrication of the object in question. The new premise to work on would be "Analyse the artefact, characterise it and then with every manufacturing process characteristics, query those of the artefact for a match", which in other words means, "take the processes to the artefact."

In a recent paper by Chen et al [211], the extraction of geometric characteristics for moulding product design assessment was analysed. Their work is based on the predefined parameters available in feature-based design, where there is a set of rules of accessible forms for tasks from design to manufacturing. The geometric characteristics, this paper works on, are those formed by feature interactions and generalised as significant items, such as "depth", "thickness", "height" and so forth.

It can be seen that although some of the technologies for feature recognition are readily available, the approach in this thesis is somewhat different. It is founded on the concept of using the characteristics of manufacturing process to interrogate the model. Because artefacts are designed to fulfil functional requirements (functional features) or assemblability constraints (assembly features), the manufacturing process capable of fabricating such artefact is usually a by-product. Therefore, any sort of knowledge available on the manufacturing process can be used to question the component and determine how suitable the fabrication technique is. In the event that the manufacturing process at hand is incapable of fabricating the artefact, then the object needs to be modified to meet the manufacturing process requirements, but it has to stay under the constraints of assembly and functional requirements.

8.2.3.3 Process characteristics – example

Every manufacturing process has a set of strong and weak points. These so called process characteristics or design aspects are well known for most manufacturing techniques. Table 8.1 and Table 8.2 present a list of design aspects (adapted from Swift and Booker [184]). These aspects can be used to interrogate and be compared with the information extracted from the artefact's geometry.

Table 8.1 - Shell moulding process properties

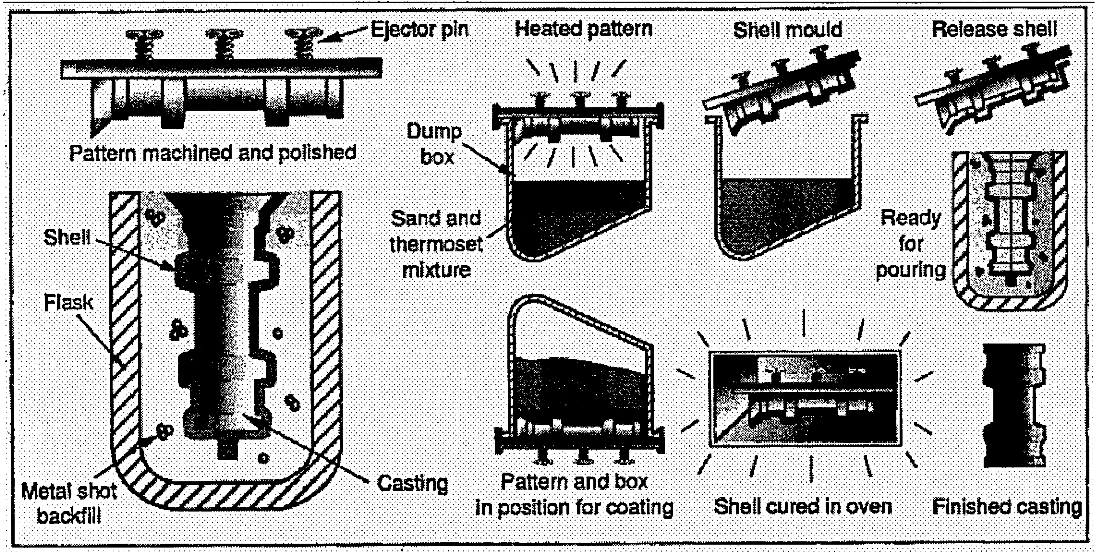


Figure 8.11 – Shell moulding process

Main characteristics (Design aspects)	Value
Draft angle range (section dependant)	0.25 – 1 degrees
Maximum section	50 mm
Minimum section	1.5 mm
Cross section variations, susceptible	FALSE
Size range	10 g - 100 kg
Optimal size	< 20 kg
Core holes	> 3 mm
Bosses	TRUE
Inserts	TRUE
Undercuts, possible	FALSE
Parting line positioning importance	TRUE
Surface roughness	0.8 – 12.5 μm Ra

This list presents constraints such as maximum and minimum section, the size/weight which this process has optimum results and so forth. Other types of information for immediate assessment are those related to the convenience of introducing features such as bosses, undercuts and parting line positioning, amongst others.

This information is readily available for most manufacturing processes and it can be incorporated in database for later extraction and comparison with that of the model being examined. This information can also be used as high-level process selection. Many real-world parts are constructed with several manufacturing processes, e.g. cast pieces that are then machined to final shape. Although this is what process planning methodologies seek to address, process characterisation can be used as a basis for either net-shape process selection or primary process preference.

Table 8.2 – Powder metallurgy process

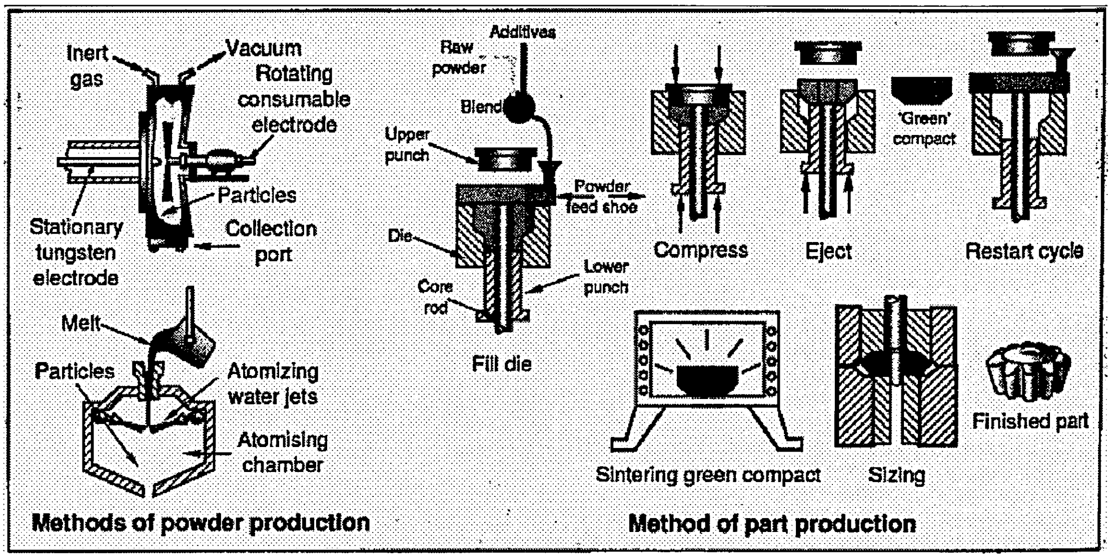


Figure 8.12 - Powder metallurgy process

Main characteristics (Design aspects)	Value
Draft angles	≥ 0 degrees
Maximum length to wall thickness ration	8 : 1
Maximum length to diameter ratio	4 : 1
Minimum section	≥ 0.4 mm
Minimum section (optimum)	1.5 mm
Cross section variations, susceptible	TRUE
Size range (weight)	10 g – 15 kg
Size range (projected area)	4 mm2 – 0.016 m2
Near-net shapes	TRUE
Preferred envelope	CYLINDRICAL
Narrow slots, susceptible	TRUE
Undercuts, possible	TRUE
Undercuts, restriction	Parallel to compaction direction

8.3 Process Handling / Manipulation complexity

The second and final sub-division of component complexity assessment is 'manipulation complexity' or process handling evaluation. As previously mentioned in chapter 4, current DFA [11, 96, 212] methodologies offer a scoring system to evaluate aspects of component manipulation (either manual or automatic) such as: component size and weight, handling difficulties, part orientation, insertion process (part placing process, fastening process, and restricted access, amongst others). Some of the charts used in the scoring system proposed by the Lucas DFA methodology are presented in Appendix B.

Understandably, this scoring system is heavily dependent on the component's geometric information. It can only be applied at the last stages of the detailing phase of the of product development cycle. However, it complements and serves as an additional means of evaluation the component suitability and complexity.

Chapter 9

FINAL DISCUSSION AND FUTURE WORK

One of the reasons for studying Product Complexity Metrics is the exploration (and exploitation) of the increasing availability in computer power and the growing in interest in developing CAE applications linked with Product Lifecycle Management (PLM) and Product Data Management (PDM) capabilities. So far, the few complexity metrics that have been presented would hopefully pave the way to develop powerful design support systems. The application of these metrics will also help to implement best design practices such as Assembly-Oriented Design.

The tools presented in this work have been oriented to the study of product architecture, where interactions and interfaces have been the centre of analysis. Nonetheless, product functionality evaluation has not been explored as it still remains an elusive concept. Furthermore, definitions of 'product functionality' are still open to debate.

Research on new complexity-management techniques is still in progress, as it is on the creation of new tools and practical new means to deal with complexity, as opposed to trying to eliminate it altogether. The very nature of 'product complexity management' is undoubtedly abstract. It is clear that any attempts to create a measure of it, let alone definitions, are certain to follow the same path as definitions of product functionality. However, within certain restrictions, the implementation of any complexity metrics, as those presented here, would certainly improve the design and development of products.

The theory was (and still is) that by achieving superior designs early in the development process, it is anticipated that the end product would be improved and the cost and time required for validation reduced. The following sections present additional factors that need be addressed for a successful creation of complexity metrics.

9.1 Complexity estimation across multiple product domains

Throughout the development of this thesis the concept of complexity has been stated as being entirely "context dependent." This notion becomes yet more evident when there is still an elusive notion: *Product application dependency*. This situation occurs despite the introduction of indicators or monitoring systems for the several types of factors that make the overall product complexity.

It follows that there are additional factors that influence the construction of a product. Some of these factors will depend exclusively on the area of application of the artefact. Consequently, the area of application can be a deciding factor in setting the difference between two products with almost identical characteristics. This factor will be referred to as "product domain."

In order to understand what is meant by 'product domain', one can study the classifications of CAD models as suggested by Shikhare [60] (see Figure 9.1), and afterwards take a similar approach in the context of product complexity. Following Shikhare's methodology, a set of CAD models has been mapped on to a grid according to their '*combinational complexity vs. geometric complexity*' behaviour. Geometric Complexity deals with the number of lines, curves, planes, surfaces, amongst others, whereas Combinational Complexity accounts for the number of components, faces and edges. As a result, if all CAD models share similar characteristics, then the complexity metrics could be applied and used for scoring purposes. It is worth mentioning that Shikhare's classification method was primarily based on the size of the CAD models.

As mentioned above, a similar principle can be used as the foundation for the sorting of products, bearing in mind that those artefacts are designed for, and within, a specific industry type that bear common traits. In this case, the difference between the classification suggested by Shikhare for CAD models and the one proposed here for the products is that, the latter deals with products not only CAD models, but also with their application domain. This additional constraint does not lend itself to be mapped in an area where one can move freely, because the classification of products does not depend exclusively on their size, making this classification more restrictive. However, the arrangement is still based on product commonality. Similarities in the artefacts make their comparison possible within the same domain (industry area) and also make the evaluation of complexity a more practical issue.

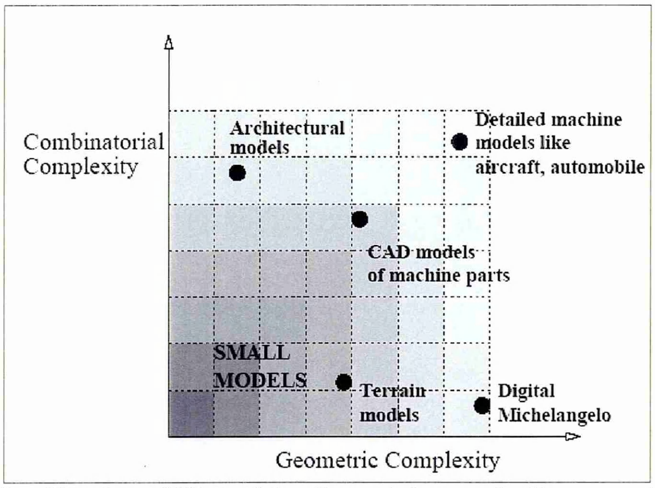


Figure 9.1 – Classification of 3D models: Geometric complexity versus Combinatorial complexity (*taken from Shikhare [60]*)

As shown in Figure 9.2, the various product ranges can be mapped into a grid of '*product-ranges vs. complexity level'*. (The graph is shown for illustration purposes only).

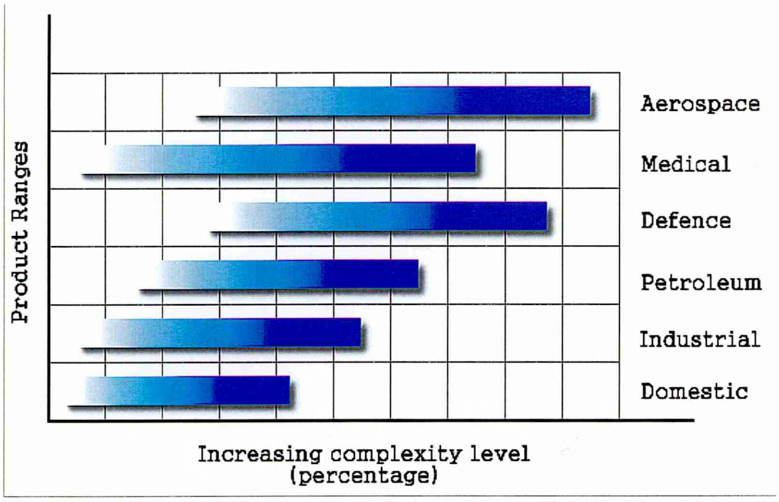


Figure 9.2 – Product complexity across application domains

What are the benefits of having different "interpretations of complexity" for different product domains? One good reason is that complexity metrics must be in accordance with the designer's knowledge and common sense. Software developed for product complexity evaluation needs to let the user know whether the product being created is complex or not, based on the area of application of the device. For instance, designers of "washing machines" would not like to end up following a certain number of guidelines only to find out that their product is less complex than a "car." This is where common sense needs to be used! The difference between cars and washing machines can be easily stated by humans, but how is a piece of software supposed to tell the difference between a moderately complex car and a considerably complex washing machine? What sort of reasoning is required to distinguish between the two? What are the leading variables? One cannot simply count the number of components or trace the number of relationships between those components. These are issues that still need to be addressed in future work. The following sections present, nonetheless, a series of ideas that my help in this endeavour.

9.2 Product domains: definitions and differentiations

There are relatively few companies dealing with a large range of diverse products, or at least, they do not manufacture them in the same shopfloor. Therefore, for the vast majority of businesses, product range definition hardly matters. However, complexity metrics need to take into account these domains. They must be sufficiently explicit to make product comparison possible within the same application level (e.g. comparison of two different low-voltage electric motors), but these metrics must also be as wide-ranging as possible to be applicable in every different industry (e.g. automotive, oil-exploration and so on). As previously declared, complexity metrics are strongly dependent on the context in which they are used.

How can one discriminate between product domains? Data such as geometry, number of components, and number of interactions does not convey the necessary amount of information to do so. It is necessary to know about types of materials, weights, product lifespan, safety factors, surface finishing and component sizes (somehow linked to geometric data) and perhaps regulating codes. This information has not been considered in any of the metrics suggested so far, yet they play a vital role in placing the designed product in a specific environment. On the other hand, if the metrics were to be used for a specific application, they would not require further alterations or enhancements.

The metrics presented here can be applied for comparison of products within the same domain, but it makes no sense to apply them to products within different domains. Although these metrics could, in theory, be applied to products from different domains, such comparison is for all purposes impractical. Comparison of products across domains does not give valuable information. For instance, comparing the complexities of a control system in washing machine with those of a control system in an airplane does not help to the assessment of control system. Both control systems have different areas of application, they belong to different domains. It does make sense, however, to compare two control systems within the aerospace domain, or two control systems within the domestic appliances field.

9.3 Complexity management and manufacturing barriers

In addition to the importance of considering discrimination between product domains, real complexity evaluation also needs to take into account the current situation of and the resources available to the company that will be implementing the suggested complexity metrics. In previous chapters, complexity evaluation was been divided in two classifications: *static* and *dynamic complexity*. The former (static complexity) is concerned with the set of factors that influence the structure of the product. It also take care of aspects such as component overall shape, component-to-component interface geometry, parallel processing capabilities, maintenance and so on. Conversely, 'dynamic complexity' involves all operational characteristics, as well as the degrees of freedom (assembly), component handling, insertion operations, and turnovers, to name but a few.

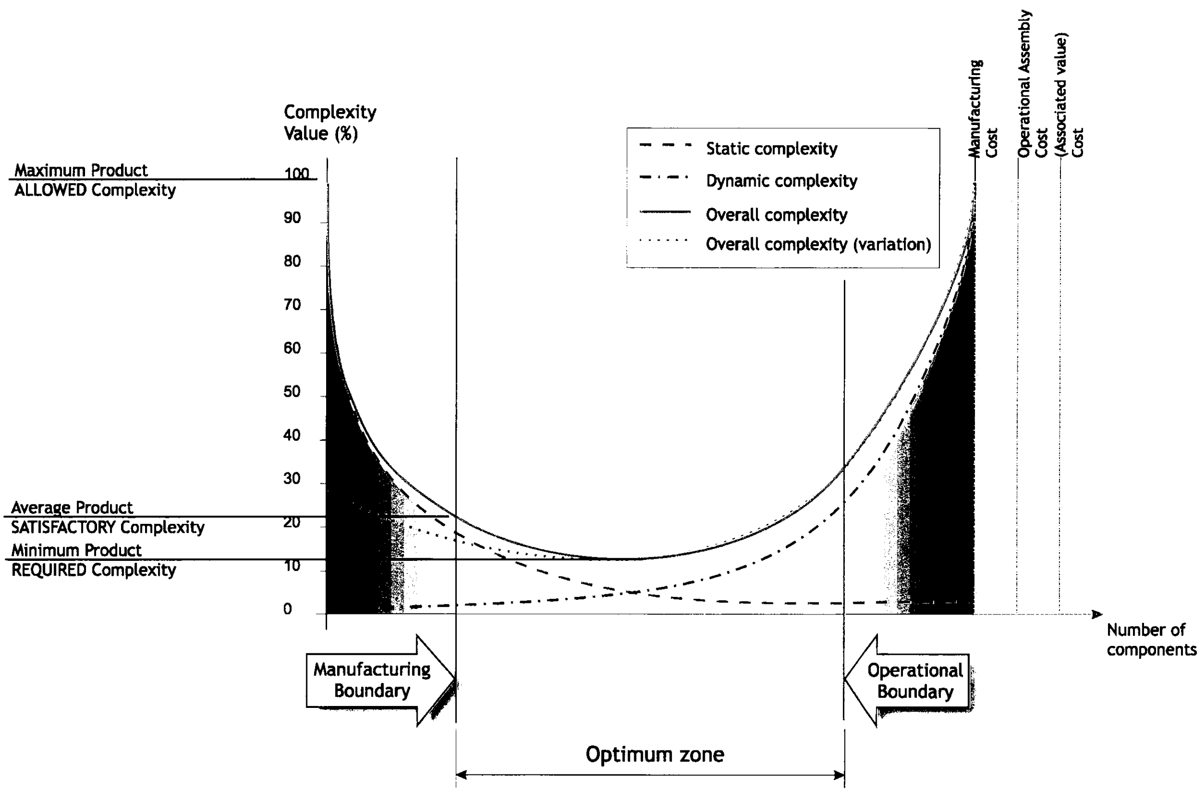


Figure 9.3 – General overview of Complexity vs. Number of parts

Figure 9.3 represents these sub-divisions of complexity (i.e. static and dynamic). For any given product, there is an inherent value of complexity, which can be manipulated depending on the analysis imposed. Subsequently, reducing the number of parts will ultimately drop the "overall complexity value" down to a certain level, from there onwards, the more parts that are eliminated (i.e. eliminated or integrated), the more complicated the remaining parts will have to become to be able to accomplish the same '*functionality*'. From the same figure, the following assumptions are extracted.

- For any given plant, the capacity for producing geometrically complex parts must have a boundary, which, in due course, will limit the degree of shape complexity of a component. This manufacturing boundary depends on the tooling and machinery available within the company (see chapter 8, where an algorithm based on manufacturing process characteristics is used to analyse the components manufacturability). (*Manufacturing boundary*)
- On the other hand, the number of components could be reduced in order to increase the assembly (or design) efficiency [11, 213]. However, in some cases this reduction might not be enough. For instance, If the hardware available in a

plant to assemble the product is not capable of meeting the manufacturing rates planned for the product current configuration, then this lack of appropriate resources will alert the designers of an overly complex product structure. In other words, the fact that a company optimises a product does not guarantee that the plant will be able to manufacture it. The plant can even implement the most convenient type of assembly (manual or automatic). If the company does not have the appropriate equipment, then the assembly is still too "complicated" (i.e. slow or demanding). The tooling available for operational purposes will end up setting up a limit to the plant productivity. If this limit were considered as the operational boundary of the plant, then a suitable complexity value for the product to accommodate this deficiency would produce a 'break-even' point for the plant production rate. (*Operational boundary*)

- The term 'overall complexity variation' is due to the use of 'off-the-shelf' components. This suggests a reduction in the overall complexity by redirecting or handing over the manufacturing of components to a different plant (i.e. another company specialised in this type of components). This will certainly drop the overall complexity even more (overall complexity variation), for it "eliminates" factors that contribute to the static complexity of a product. This practice will ultimately have a strong influence in the product cost. New components can be overly complex geometrically, but since the company is not manufacturing those components, then the plant is released from any manufacturing costs attributable to geometric/shape complexity. This could encourage part variation reduction and standardisation for 'outsourcing'.

The zone between 'manufacturing boundary' and 'operational boundary' represents the optimum zone for product complexity. Every product has a built-in level of complexity (numerical; low, medium, or high) such value will be reflected on the product cost, reliability, service, and ergonomics, amongst others.

9.4 Relationship between product domains (general values) and manufacturing barriers (specific values)

Referring back to the product complexity ranges definitions (previous section), this zone of optimum product complexity is linked to the chart that represents the different complexity values for a given product (see Figure 9.2). From

Figure 9.3, the limit labelled as 'maximum *allowed* product complexity value" represents the 100% of product complexity, but in Figure 9.2, this value represents the left most end of the product complexity for that particular area of application.

For the sake of illustration, consider the domain for medical products. Their overall complexity value fluctuates[39] between 30 – 95% (see Figure 9.3). This represents the expected complexity value in a general product scale. However, the top value of 95% in the global scale will be equivalent to the "maximum *allowed* product complexity value" in the local scale (i.e. locally a 100% product complexity). Consequently, the 30% in the global scale will be equivalent to "the minimum *required* product complexity value." In other words, Figure 9.3 represents the measure of specific product complexity, and if the product needs to be compared with products from different domains, then this value needs to be translated back to the global scale, in chart similar to Figure 9.2.

In addition to, maximum (*allowed*) and minimum (*required*) product complexity, there is the "*average-and-satisfactory* product complexity value". This value is not only product dependent, but also 'plant dependent'. The manufacturing and/or operational boundaries will dictate such an average value. In the case of Figure 9.3, manufacturing boundary is the one controlling such estimation.

The previous analysis adds another twist to the concept of context-based complexity estimation. It simply declares that product development and design complexity not only depends on the type of product being designed, but on the organisation or plant fabricates it. Measures of complexity for a specific plant can help to compare and reduce unnecessary costs within the plant. This can be achieved by improving the design or the manufacturing process or the product handling or all of them at the same time, therefore constructing a product suitable for that specific plant. In this manner, the company will be able produce an artefact that is relatively easy to manufacture, but that might be or not be more complex in comparison with other organisations producing the same artefact. This argument raises a new question about complexity management: *if an organisation is comfortable with the products it creates, do they need to worry about making their products look more complex than*

[39] Hypothetical example, these values are presented for the illustration purposes only, since no real values have been found yet.

those of its competitors? Could a company benefit from having a product that appears complex to others? The answer is yes!

From the discussion presented above, one could infer that once a company has found the simplest solution to manufacturing a product, they could step back and make it give the impression of being more complex to the outside world. In this case, if the company is comfortable with the performance of the systems that manufactures the product and the product is well suited to their equipment, then understanding and coping with the complexity of the product could benefit the company. Product complexity could be used as means to encode and disguise the product architecture and thus protect the company from reverse engineering or piracy.

9.5 Complexity metrics and decision support integration

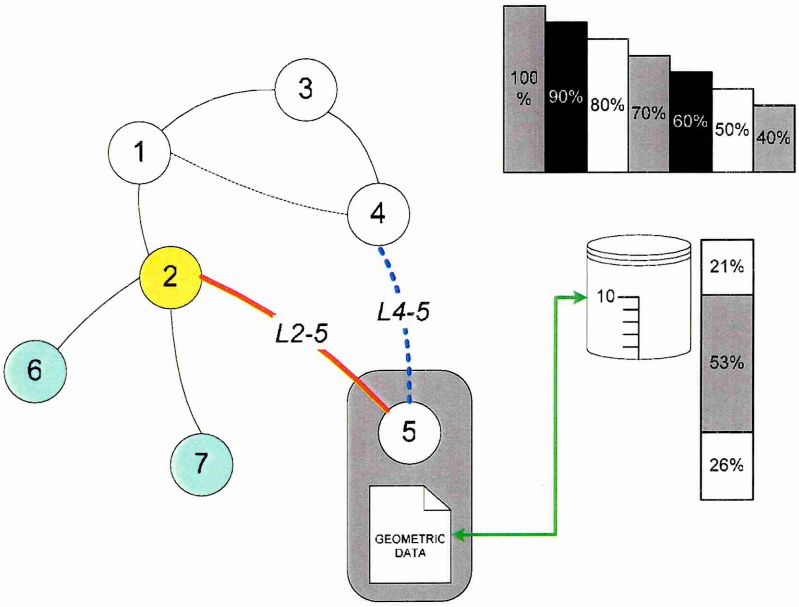


Figure 9.4 – Complexity metrics and decision support integration

As expected the integration of complexity metrics within a decision support system has to be as smooth as possible. This is part of the further work implied by this thesis. This implementation and integration is intended to be running in the background of a design environment system, thus avoiding causing any possible disturbance to the user and, consequently, being available only when necessary or on demand from the user. That is, the user can choose whether to run or not this particular evaluation module whilst using the software application that contains it.

It has been seen elsewhere in this work, that the metrics depend on the product, company and design stage at which they are needed. For that reason, advice in certain areas would only be accessible as information becomes available. For instance, advice in areas such as manufacturing viability (shape complexity) would only be possible once geometric data is created - no difference with conventional CAD packages there. However, since there is a strong emphasis in Assembly-Oriented design, geometric information for a particular component is dependent on the links that it has with other components.

Component connectivity has been the centre of the analysis presented in this work, and it involves the evaluation of product structure. When two components interact, the intensity of their interaction and the number of total interactions are some of the variables recorded during the design process.

A rough schematic approach is depicted in Figure 9.4. At the concept design stage, certain components have been defined and a product structure is outlined. In this example, Component No. 2 (C2) registers a high number of interactions; usually, such characteristic is typical of a base part. This could be pointed out later on in the development process, perhaps during assembly planning. Component No 6 (C6) and 7 (C7) relate only to one other component (coincidentally to C2). Such lack of interaction suggests that any modification made to their geometric shape will not have a greater implication in the structure of the product. Any modifications to either C6 or C7 will just have and impact on connecting partner (C2). Component No 5 (C5) has two different types of interactions with two other components (C2 and C4). One of C5's interactions could be a strong geometrical interaction. This geometric interaction will partially determine its overall geometric shape. Additionally, C5 has another non-geometric interaction, possibly, exclusively due to adjacency. In chapter 6, it was explained that interactions can be of a combined nature (geometric and adjacent), however one type of interaction could be more dominant than the other, which clearly controls the type of interface involved.

The type of interfaces available in C5 (C5-C2, and C5 - C4) could also be further analysed for possible product modularisation or energy exchange (as presented by Holtta [140] and described in chapter 4). Product modularisation or modularity analysis also influences the way in which the components can be fabricated and assembled [113], that is, they could be manufacture in parallel or dependent of others. (This is contained within the structural complexity estimation).

Finally and as mentioned above, as geometric data becomes available, a manufacturability analysis can be carried out in the background. It could suggest a possible redesign of its shape for a complete fabrication-process to component-shape match. (Shape complexity). These geometric data also allows part handling evaluation using already tested geometric reasoning algorithms [174, 175] and existing scoring systems for this purpose [11].

Chapter 10

CONCLUDING REMARKS

During this research project, it was found that the Designers' Sandpit would benefit from a system (or software module) that could, amongst other things, analyse, detect and monitor the formation of patterns in the product structure for an objective and proactive implementation of DFA. It was also found that, in order to create and implement such a monitoring system, the complexity of the product needed to be analysed. Consequently, it was suggested that the best way to achieve such a goal would be through the extraction of complexity metrics that would help assess the product structure.

To facilitate the establishment of suitable metrics, it was necessary to explore the literature for ideas, first to find out whether such metrics could be produced. This survey would serve as a starting place to learn whether there were any existing and appropriate metrics that could be used, or integrated, into the project.

A large literature survey quickly revealed that the concept of complexity was extremely broad and that it covered too many different areas (i.e. not only engineering). Furthermore, the studied literature also pointed out that complexity, as a concept, was hard to define and that it was context dependent. After an observation of the different notions of complexity, it was found that complexity has been explored more intensively in the studies of complex systems (i.e. biological systems, manufacturing systems, or social systems), but little has been written about complexity in the design of products, or even about the structure of the products. More interestingly, when product design complexity had been investigated, the most common type of complexity assessment found in the literature was qualitative rather than quantitative. The latter finding could have suggested that complexity metrics could not be product appropriately, hence the lack of information about them.

However, some metrics have been successfully produced for assembly sequence complexity, component handling/manipulation (already available as part of a scoring system in most DFA methodologies), product modularity metrics and even some shape complexity metrics (reminiscent of part coding). Nevertheless, for the purposes of proactive DFA implementation, these metrics not only have to be integrated, but new ones had to be produced. For instance, product architecture has been neglected, therefore, binary component interaction and component-to-component connection interfaces have not be explored within the context of product design, and even less within the context of DFA implementation.

This thesis successfully produced a set of metrics that can be integrated and evaluated within a product design and development context. It effectively managed to present a way to integrating existing metrics (discovered through a large survey of the published literature) with newly development metrics, through the proposal of appropriate product complexity taxonomy.

The study of assembly complexity has been an essential and central part of this work. This research managed to explore and make emphasis on the overlooked and essential area of 'assembly structure complexity'. Additionally, it suggested a novel way of improving on the assessment of component shape complexity, allowing the integration of characterisation of manufacturing processes with the evaluation of components for manufacturability.

Moreover, this work managed to explore and state the benefits of applying and using these complexity metrics in the product development cycle. However, it has also produced more intriguing questions, such as the creation of metrics for functionality assessment or metrics for evaluation of product within and across application domains. Unfortunately, these last issues have been left as part of a further work to complement this study, but it promises to be as interesting and engaging as this entire work has been.

The remainder of this chapter presents a review of the lessons learnt during this research project.

10.1 The important of product complexity assessment

This dissertation has presented a few of the complexity metrics that can be inferred from a description of the product development cycle. This work can be used to hopefully pave the way for the development of more powerful design and decision-making support systems, systems that are being created to implement best design and engineering practices. Furthermore, the title of this thesis indicates that the evaluation of product complexity must be integrated in CAE applications that seek to implement Product Lifecycle Management (PLM) and Product Data Management (PDM) capabilities. The evaluation of product complexity is not only gaining currency, but it is also becoming increasingly necessary to understand the management of production resources, since they are more limited and timescales for the development new (or optimisation of) products need to be reduced in order to compete in the current business environment.

10.2 Information extracted from the case studies

Examination of the cases studies showed that valuable information can be extracted even with the reduced implementation of the overall analysis suggested in this work. Product configuration (layout) and a little prior knowledge of the types of interfaces between pairwise components were sufficient to indicate possible consequences downstream in the development cycle. Additionally, it was discovered that variables such as *connectivity, cohesion levels, component variation* and *amount/type of interface* do indicate situations addressed by DFA methodologies. Usually, the metrics prompted for modifications that could have possible future possible consequences (in the redesigns). More often than not, the situations detected were addressed, corrected and optimised later on in the suggested redesigns.

Furthermore, comparing originals with suggested redesigns, one can see that DFA-optimised products tend to follow specific patterns. This evolutionary behaviour was identified after finding and analysing the following phenomena:

1. After the re-design of components and their assembly, re-designed products often change their architectural configuration. They can no longer be represented by a rigid hierarchical tree-like arrangement where ramifications of the product

structure makes possible to identify independent[40] sub-assemblies. The newly designed products become more compact and, for that matter, components are more interconnected. It can be said that the architecture of the product evolves from a tree-like layout into a lattice-like configuration (see Figure 10.1)

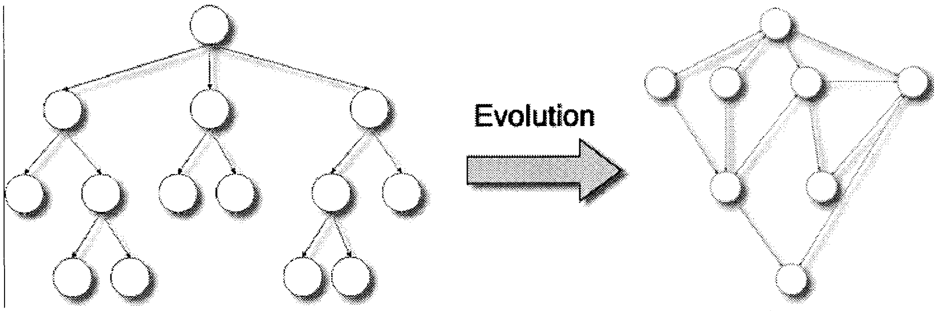


Figure 10.1 – DFA-optimised product structure's evolution

2. The product structure evolves into one with higher cohesion levels. These high cohesion levels generally mean that components are performing essential functions and that they interact with their neighbours to transmit this functionality. Loose configurations generally highlight tree-like (hierarchical) architectures, typical of complex systems, where sub-systems are designed to give support to other components of the product.
3. High cohesion levels also indicate that modifications to any component tend to have a greater and, furthermore, more general impact on other components' behaviour and shape (modifications spread throughout the structure). This can be addressed by assembly-oriented design environments, where interfaces between any two pairs of components are defined and accounted for, as well as the component internal geometry due to the task it needs to perform. Assembly-oriented designs can "absorb" these modifications more easily than component-oriented designs. The latter, on the other hand, require a complete redesign of the affected components which is often reflected in lengthy and trial-and-error optimisation operations.
4. Geometric awareness. Non-geometric interfaces cut the flow of neighbourhood awareness. Components with dominant non-geometric interfaces are less likely to

[40] This independence is not entirely true because components are still put together; this adjacency seems more at random than well-thought interaction between any two pair of components.

be assembled easily, as their geometric shape (manufacture) does not keep any link with other components.

5. Finally, the identification of vertical assemblability and/or turnover operations during complexity assessment simply follows typical DFA guidelines.

10.3Lessons learned from the complexity metrics study

This work has in some ways raised more questions that it actually answered. However, it has demonstrated that the metrics suggested can effectively draw attention to the different issues that need to be monitored during the design process. This information helps the designers to cope with the several and ever present factors affecting and sources of complexity. It has also shown that complexity in a product cannot be eliminated, but must rather be managed - thus requiring tools and means to do so.

The following observations have been extracted from the work in general, they are not intended to be taken as a brief summary, but rather as signposts that serve to highlight the main characteristics of complexity in products and product design.

- Complexity metrics (CMs) help to control the design process and draw attention to critical changes.
- CMs eliminate subjective estimations and help detection of most cost-effective product architecture.
- CMs can and must be produced for the different product stages (concept design, detailed design, construction, system use),
- Any sort of complexity metrics extracted, although generalised, must also be tailored to suit the organisation's available resources (organisation specific),
- Following the previous statement, Complexity metrics must be dependent on product area: medical, industrial, petroleum, aerospace, etc. (application specific)
- Complexity metrics are and must be subjective across various product application areas, but must be objective within the same area (organisations are tested for their feasibility to create any given product),

- Finally, the complexity metrics proposed in this work are basically for the detailed design stage of the product, and should be complemented with additional work.

One of the aims of drawing up complexity metrics is the elimination of subjective estimations. This allows an entire and replicable method of detecting which product configuration is the most efficient (from the DFA viewpoint) and cost effective. Moreover, these metrics will not only help to control the design process (e.g. constantly drawing attention to critical changes), but they will also help to compare two product configurations of the same design. Such comparisons will ultimately point out which architecture requires less design effort and lower production costs.

Notice that the terms *product* and *system* have sometimes been used interchangeably throughout this thesis. Acknowledging that, a 'product' can be sometimes regarded as 'a system' accounts for the variations in considerations given to the term 'part'. Thus, a 'part' could be a 'component' or a 'sub-assembly'. Take, for instance, a 'car door' (and all the elements inside). Some designers/manufacturers can take it as 'one part', therefore evaluation its handling and insertion characteristics as a whole. On the other hand, designers can think of it as a 'sub-assembly'. Moreover, a 'handle' –from the same door– could be considered as 'one part' belonging to a 'door' (sub-assembly), hence taken as a 'component.' These interpretations were discussed in chapter 3 and 4, which were dedicated to study the differences and implications of product complexity, system complexity, and complex systems.

There is however an important characteristic of what is generally classified as a product. A product, unlike a part, can be considered as the nucleus of the system. Therefore, it does not matter much if a product *is* a system or it *belongs to* a system, in this case, what really matters is that the evaluation of a product will raise an interesting question: *Is the product leading the complexity of the production system? Or is there a mutual constraint?* The reason for this question, amongst others, is the need to estimate the time and effort required to manufacture the product, as well as the costs involved in building and planning a whole process around it (the system).

This thesis has presented a framework for the estimation of complexity metrics within a proactive implementation of DFX methodologies. It has mainly focused on metrics for the detailed design stage of the product, where at least a first approach to the product architecture has already been considered. Such metrics are intended for use in conjunction with those drawn up for the different product stages (concept design, detailed design, construction and system use). Although the concept of complexity implies non-linear estimations, it is known that recording historical data will serve as the 'know-how' for predicting the repercussions for product construction and use, as the product architecture changes throughout the design process.

The author is aware that there are still shortfalls in the analysis proposed, for example, it has not given sufficient information about the functional decomposition of the product. This additional analysis is left as supplementary work to be carried out in the future.

In spite of that, this thesis has managed to confirm one of the most important aspects of the metrics required for complexity analysis, namely, that the assessment of complexity as such cannot be reduced to a single number. The very nature of the concept requires a multi-dimensional estimation that needs to be complemented with other metrics, such as: *formalisation* (from concept to detailing), *functionality*, *modularity* and *product viability*, to name a few. This is part of an ongoing research process, in the scientific community, which has promised to be as complex and the subject it studies!

References

1. Thallemer, A., 2004. *Bionic Design - The human touch of technology.* in *Proceedings of the 8th International Design Conference DESIGN 2004.* Dubrovnik, Croatia: Designer Society.
2. Kim, B.S., 1999. *Designing for Simplicity: Lucent Technologies uses a method called Design for Simplicity to improve the assembly of equipment that houses and connects the company's telecommunication products.* Mechanical Engineering, **121**(Supp/1): pp. 34-36.
3. Ulrich, K., 1995. *The role of product architecture in the manufacturing firm.* Research Policy, **24**(3): pp. 419-440.
4. Ulrich, K.T. and Eppinger, S.D., 2004, *Product design and development.* 3rd ed. Boston: McGraw-Hill/Irwin. xviii, 366 p.
5. Tate, S.J., Jared, G.E.M., Brown, N.J., and Swift, K.G., 2000. *DETC2000/DFM-14014 An Introduction to the Designers' Sandpit.* in *DFM2000 Design for manufacturing conference.* Baltimore, MD: ASME.
6. Designers' Sandpit Project, 2004, *The Designers' Sandpit Project: An Assembly-Oriented Design Environment (website).* Online: http://www.eng.hull.ac.uk/research/sandpit/, Accessed on: 17.July.2004
7. Tang, V. and Salminen, V.K., 2001. *Towards a theory of complicatedness - framework for complex systems analysis and design.* in *International conference on engineering design.* Glasgow: Bury St Edmunds - The Design Society.
8. Dalgleish, G.F., Swift, K.G., Barnes, C.J., and Jared, G.E.M., 1998. *Computer Support for Proactive DFA.* in *Tools and methods for concurrent engineering.* Manchester: [np].
9. Barnes, C., Dalgleish, G., Jared, G.E.M., and Swift, K.G., 1998. *Support For Design Evaluation In An Assembly Oriented CAD Environment.* in *Proceedings Of Engineering Design and Automation.*
10. Barnes, C., Dalgleish, G., Jared, G.E.M., and Swift, K.G., 1999. *Assembly Oriented Design.* in *IEEE Symposium on Assembly and Task Planning (ISATP 99).* Porto, Portugal.
11. Lucas Engineering Systems Ltd. and University of Hull, 1996, *Design for Assembly/Manufacture Analysis Practitioners Manual.* Version 10.5 ed. Solihull, UK: CSC Manufacturing.
12. Sturges, R.H. and Kilani, M.I., 1992. *Towards an integrated design for an assembly evaluation and reasoning system.* Computer-Aided Design, **24**(2): pp. 67-79.
13. Kim, G., Lee, S., and Bekey, G., 1995. *Comparative Assembly Planning During Assembly Design.* in *Proceedings IEEE International Symposium on Assembly and Task Planning.*
14. Barnes, C., Dalgleish, G., Jared, G.E.M., and Swift, K.G., 1997. *Assembly Sequence Structures In Design For Assembly.* in *Proceedings IEEE International Symposium on Assembly And Task Planning*: IEEE.

15. Corning, P.A., 1998. *Complexity Is Just a Word!* Technological Forecasting and Social Change, **59**(2): pp. 197-200.
16. Edmonds, B., 1997, *A Hypertext Bibliography of Measures of Complexity*. Online: http://bruce.edmonds.name/combib/, Accessed on: 28.Jan.2004
17. Edmonds, B., 2000. *Complexity and Scientific Modelling.* Foundations of Science, **5**(3): pp. 379-390.
18. Simon, H.A., 1996, *The sciences of the artificial.* 3rd ed. Cambridge, Mass.: MIT Press. xiv, 231 p.
19. McCarthy, I.P., Rakotobe-Joel, T., and Frizelle, G., 2000. *Complex systems theory: Implications and promises for manufacturing organizations.* Int. J. Manufacturing Technology and Management, **2**(1-7): pp. 559 - 579.
20. Jeffrey, P., Allen, P., Seaton, R., and Thomson, A., 2000. *Complex systems research as the structuring of cross-disciplinary knowledge*. in *International transdisciplinarity conference; Transdisciplinarity*. Zurich, Switzerland: Haffmans Sachbuch.
21. Agudelo-Murguía, G. and Alcalá-Rivero, J.G., 2003, *La Complejidad.* Online: http://www.iieh.com/doc/doc200301110300.html, Accessed on: 15.Oct.2003
22. Hatch, M.J. and Tsoukas, H., 1997. *Complex Thinking About Organizational Complexity: The Appeal of A Narrative Approach to Complexity Theory.* Warwick Business School Research Papers: pp. ALL.
23. Ottino, J.M., 2004. *Engineering complex systems.* Nature, **427**(January): pp. page 399.
24. Frizelle, G., 1998, *The Management of Complexity in Manufacturing: A Strategic Route Map to Competitive Advantage through the Control and Measurement of Complexity*. London: Business Intelligence. 320.
25. Kauffmann, S., 1996, *At Home in the Universe: The Search for Laws of Self-Organization and Complexity*. Oxford: Oxford University Press. 336.
26. Smart, J.A., 2003. *On the application of irreducible complexity.* Progress in Complexity, Information and Design - The Journal of ISCID, **2.1 and 2.2**(Jan - June): pp. 11.
27. Kauffman, S.A., 1993, *The origins of order: self-organization and selection in evolution*. New York: Oxford University Press. xviii, 709 p.
28. Waldrop, M.M., 1992, *Complexity: the emerging science at the edge of order and chaos*. New York: Simon & Schuster. 380 p.
29. Heylighen, F., 1989. *Self-organization, Emergence and the Architecture of Complexity*. in *Proceedings of the 1st European Conference on System Science*. Paris.
30. Rakotobe-Joel, T., McCarthy, I.P., and Tranfield, D., 2002. *A Structural and Evolutionary Approach to Change Management.* Computational and Mathematical Organization Theory, **8**(4): pp. 337-364.
31. Wooldridge, M.J., 2002, *An introduction to multiagent systems*. New York: J. Wiley. xviii, 348 p.
32. McCarthy, I.P., 2002. *Manufacturing fitness and NK models*. in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.
33. Adamides, E.D., 2002. *The use of fitness landscape theory and systems dynamics for the development of manufacturing strategy*. in *2nd Internal.*

Conf of the Manufacturing complexity network. University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.

34. Assogna, P., 2002. *Knowledge management and fitness landscape.* in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.
35. Kauffman, S.A., 1996, *At Home in the Universe: The Search for Laws of Self-Organization and Complexity.* Oxford: Oxford University Press. 336.
36. Shannon, C.E., 1948. *A Mathematical Theory of Communication.* The Bell System Technical Journal, **27**(July, October): pp. 379-423, 623-656.
37. Shannon, C.E. and Weaver, W., 1999, *The Mathematical Theory of Communication.* 50th ed ed. Illinois: University of Illinois Press.
38. Frizelle, G. and Woodcock, E., 1995. *Measuring complexity as an aid to developing operational strategy.* International Journal of Operations and Production Management, **15**(5): pp. 26.
39. Calinescu, A., Efstathiou, J., Sivadasan, S., Schirn, J., and Huaccho-Huatucco, L., 2000. *Complexity in Manufacturing: An information theoretic approach.* in *Intl. Conf. on Complex Systems and Complexity Manufacturing.* Warwick University.
40. Huaccho-Huatucco, L., Efstathiou, J., Sivadasam, S., and Calinescu, A., 2001. *A model for the control of complexity in manufacturing systems.* in *10th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2001).* Vienna, University of Technology, austria.
41. Deshmukh, A.V., Talavage, J.J., and Barash, M.M., 1998. *Complexity in manufacturing systems: Part 1 - Analysis of static complexity.* IIE Transactions, **30**(7): pp. 645-655.
42. Efstathiou, J., Calinescu, A., Schirn, J., Fjeldsoe-Nielsen, L., and Sivadasan, S., 1999. *An expert system for assessing the effectiveness of manufacturing information systems.* Advances in Manufacturing Technology, **13**: pp. 161-166.
43. Calinescu, A., Efstathiou, J., Huatuco, L.H., and Sivadasan, S., 2001. *Classes of complexity in manufacturing.* Advances in Manufacturing Technology, **15**: pp. 351-356.
44. Koestler, A., 1967, *The ghost in the machine.* London: Hutchinson. xiv, 384 p.
45. Simon, H.A., 1969, *The sciences of the artificial.* Karl Taylor Compton lectures, 1968. Cambridge: M.I.T. Press. xii, 123 p.
46. Monostori, L., 2003. *AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing.* Engineering Applications of Artificial Intelligence, **16**(4): pp. 277-291.
47. Van der Heijden, K., 1996, *Scenarios: the art of strategic conversation.* Chichester, England New York: John Wiley & Sons. xiv, 305 p.
48. Lyons, M., 2002. *Complex systems models for strategic decision making.* in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.
49. Allen, P.M., 2002. *Evolution, Emergence and Learning in Complex Systems.* in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.

50. Jones, A.T., Reeker, L.H., and Deshmurkh, A.V., 2002. *On information and performance of complex manufacturing systems*. in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.

51. Casti, J.L., 1992, *The simply complex: trendy buzzword or emerging new science*. Santa Fe Institute: Santa Fe, New Mexico, USA. pp. ALL. Bulletin. 7

52. Frenken, K. and Windrum, P., 2000. *Product differentiation and product complexity: A conceptual model and an empirical application to microcomputers*. in *8th Intl. Joseph A. Schumpeter Society Conference*. Manchester, UK.

53. Potter, J., 2002. *Leadership - Handling the human side of complexity*. in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.

54. Hirschi, N.W. and Frey, D.D., 2002. *Cognition and complexity: An experiment on the effect of coupling in parameter design.* Research in Engineering Design, **13**: pp. 123-131.

55. Bashir, H.A. and Thomson, V., 1999. *Estimating design complexity.* Journal Of Engineering Design, **10**(3): pp. 247 - 257.

56. Guenov, M.D., 2002. *Complexity and cost effectiveness measures for systems design*. in *2nd Internal. Conf of the Manufacturing complexity network.* University of Cambridge, UK: Institute for Manufacturing, University of Cambridge.

57. Suh, N.P., 1990, *The principles of design*. Oxford series on advanced manufacturing. New York: Oxford University Press. xiv, 401.

58. Shikhare, D., 2001, *Complexity of geometric models*. National Centre for Software Technology: Mumbai. pp. ALL.

59. Chazelle, B. and Incerpi, J., 1984. *Triangulation and shape complexity.* ACM Transactions on Graphics, **3**(2): pp. 135-152.

60. Shikhare, D., 2001, *State of the Art in Geometry Compression*. National Centre for Software Technology: Mumbai. pp. ALL.

61. Luebke, D., 1998, *A Survey of Polygonal Simplification Algorithms*. University of North Carolina: Chapel Hill. pp. ALL.

62. Chase, S. and Murty, P., 1999. *Evaluating CAD Complexity.* International Journal of Design Computing, **2**.

63. Kyprianou, L.K., 1980, *Shape classification in Computer-Aided Design*, University of Cambridge: Cambridge, UK. PhD Dissertation

64. Ames, A., 1988. *Automated Generation of Uniform Group Technology Part Codes From Solid Model Data*. in *Computers in Engineering*. San Francisco: ASME.

65. Love, D.M. and Barton, J.A., 2001. *Drawing retrieval using an automated coding technique*. in *Int. Conf. on Flexible Automation and Intelligent Manufacturing (FAIM)*. Dublin.

66. Rea, H.J., Corney, J.R., Clark, D.E.R., Pritchard, J., Breaks, M.L., and Macleod, R.A., 2002. *Part-sourcing in a Global Market.* Concurrent Engineering Research and Applications, **10**(4): pp. 325-334.

67. Sung, R., Rea, H.J., Corney, J.R., Clark, D.E.R., and Pritchard, J., 2002. *ShapeSifter: a retrieval system for databases of 3D engineering data.* New Review of Information Networking, **8**: pp. 33-54.
68. Corney, J., Rea, H., Clark, D., Pritchard, J., Breaks, M., and MacLeod, R., 2002. *Coarse Filters for Shape Matching.* Ieee Computer Graphics and Applications, **22**(3): pp. 65-75.
69. Corney, J.R., Clark, D.E.R., Murray, J.L., and Yue, Y., 1992. *Automatic Classification of 2 1/2D Components.* Concurrent Engineering, **59**: pp. 85.
70. Little, G., Tuttle, R., Clark, D., and Corney, J.R., 1998. *A Feature complexity index.* IMechE Journal of Mechanical Engineering Science - Part C, **212**(C5): pp. 405-413.
71. Psarra, S. and Grajewski, T., 2001. *Describing Shape and Shape Complexity Using Local Properties.* in *3rd Intl. Symposium on Space Syntax.* Georgia Inst. of Tech, GA (USA).
72. Veltkamp, R.C., 2001. *Shape matching: Similarity measure and algorithms.* in *Proceedings Shape Modelling International.* Genova, Italy: IEEE Press.
73. Tomov, B., 1999. *A new shape complexity factor.* Journal of Materials Processing Technology, (92-93): pp. 439-443.
74. Kim, G.J., Bekey, G.A., and Goldberg, K.Y., 1992. *A Shape Metric for Design-for-Assembly.* in *IEEE Conference on Robotics and Automation.* Nice, France.
75. Tate, S.J., 2000, *Symmetry detection and shape analysis for assembly-oriented CAD,* Cranfield University: Cranfield, Bedfordshire (UK). PhD Dissertation
76. Goldwasser, M. and Motwani, R., 1997. *Complexity Measures for Assembly Sequences.* International Journal of Computational Geometry and Applications, **9**(4 & 5): pp. 371 - 471.
77. Wilson, R.H. and Latombe, J.C., 1994. *Geometric Reasoning About Mechanical Assembly.* in *Algorithmic foundations of robotics.* San Francisco; CA: A K Peters.
78. Weiss, M.A., 1992, *Data structures and algorithm analysis.* Redwood City, Calif.: Benjamin/Cummings Pub. Co. xvii, 455 p.
79. McCabe, T.J., 1976. *A complexity measure.* IEEE Trans. On Software Engineering, **SE-2**(4): pp. 308-320.
80. McCabe, T.J. and Butler, C.W., 1989. *Design complexity measurement and testing.* Communications of the ACM, **32**(12): pp. 1415-1425.
81. Riguzzi, F., 1996, *A survey of software metrics.* DEIS, Università di Bologna: Bologna. Technical Report DEIS-LIA-96-010, LIA Series n.17.
82. Maimon, O. and Braha, D., 1992. *A Proof of the Complexity of Design.* Kybernetes, **21**(7): pp. 59.
83. Maimon, O. and Braha, D., 1996. *On the Complexity of the Design Synthesis Problem.* Ieee Transactions on Systems Man and Cybernetics Part a Systems and Humans, **26**(1): pp. 142-150.
84. Suh, N.P., 1999. *A Theory of complexity, periodicity and the design axioms.* Research in Engineering Design, **11**(2): pp. 116-131.
85. Suh, N.P., 1998. *Axiomatic Design Theory for Systems.* Research in Engineering Design, **10**(4): pp. 189-209.

86. El-Haik, B. and Yang, K., 1999. *The components of complexity in engineering design.* IIE Transactions, **31**(10): pp. 925-934.

87. Braha, D. and Maimon, O., 1998. *The Measurement of a Design Structural and Functional Complexity.* Ieee Transactions on Systems Man and Cybernetics Part a Systems and Humans, **28**(4): pp. 527-534.

88. Halstead, M.H., 1977, *Elements of software science.* Operating and programming systems series; 2. New York: Elsevier. xiv, 127 p.

89. Smith, C.P., 1980. *A software science analysis of programming size.* in *Proceedings of the ACM 1980 annual conference*: ACM Press.

90. Kalsi, M., Hacker, K., and Lewis, K., 2001. *A Comprehensive Robust Design Approach for Decision Trade-Offs in Complex Systems Design.* Transactions-American Society of Mechanical Engineers Journal of Mechanical Design, **123**(1): pp. 1-10.

91. Beiter, K.A., Cheldelin, B., and Ishii, K., 2000. *Assembly Quality Method: A tool in aid of product strategy, design and progress improvements.* in *ASME Design Engineering Technical Conferences.*

92. Shibata, H., Cheldelin, B., and Ishii, K., 2001. *DETC2001/DFM-21194 Assembly Quality Methodology: An Application of the Defect Prediction to Audio Equipment Assembled at Various Manufacturing Sites.* in *Design for manufacturing conference.* Pittsburgh, PA: Asme.

93. Hinckley, C.M., 1993, *A global conformance quality model: a new strategic tool for minimizing defects caused by variation, error and complexity*, Standford University: Standford, CA. PhD Dissertation

94. Hinckley, C.M., 2003. *Make no mistake-errors can be controlled.* Quality and Safety in Health Care, **12**(5): pp. 359-365.

95. Lee, B.H., Rhee, S., and Ishii, K., 1997. *Robust design for recyclability using demanufacturing complexity metrics.* in *Design Engineering Technical Conferences and Computers in Engineering.* Sacramento, CA (USA).

96. Boothroyd, G. and Dewhurst, P., 1991, *Product Design for Manufacture and Assembly*, in *Design for Manufacture: Strategies, Principles and Techniques*, J. Corbett, M. Dooner, and C. Pym, Editors. Pearson Addison Wesley. pp. 165-173.

97. Balazs, M.E., 1999, *Design Simplification by Analogical Reasoning*, Worcester Polytechnic Institute: Worcester, MA (USA). PhD Dissertation

98. Dalgleish, G.F., Jared, G.E.M., and Swift, K.G., 2000. *Design for assembly: influencing the design process.* Journal of Engineering Design, **11**(1): pp. 17-30.

99. Miles, B.L., 1989. *Design for Assembly - A key element within design for manufacture.* Journal of Automobile Engineering, **203**(D1): pp. 29 - 38.

100. Miles, B. and Swift, K.G., 1998. *Design for Manufacture and Assembly.* Manufacturing Engineer, (October): pp. 221 - 224.

101. Boothroyd, G., 1994. *Product Design for Manufacture and Assembly.* Computer Aided Design, **26**(7): pp. 505-520.

102. Richardson, M., Jones, G., Torrance, M., and Thorne, G., 2003, *Identifying the task variables that influence assembly complexity*, in *Contemporary ergonomics 2003*, P. McCabe, Editor. Taylor & Francis: London. pp. 624.

103. Whitney, D.E., Mantripragada, R., Adams, J.D., and Rhee, S.J., 1999. *Designing Assemblies.* Research in Engineering Design, **11**(4): pp. 229-253.

104. Baldwin, D., Abell, T., De Fazio, T., and Whitney, D., 1991. *An Integrated Computer Aid For Generating and Evaluating Assembly Sequences For Mechanical Products.* IEEE Transactions on Robotics and Automation, **7**(1): pp. 78-93.

105. Bullinger, H. and Ammer, E., 1984. *Computer Aided Depicting of Precedence Diagrams - A Step Towards Efficient Planning in Assembly.* Computers and Industrial Engineering, **8**(3): pp. 165-169.

106. De Fazio, T., Abell, T., Amblard, G., and Whitney, D., 1990. *Computer Aided Assembly Sequence Editing And Choice: Editing Criteria, Bases, Rules and Techniques.* in *Proceedings IEEE International Conference On Systems Engineering.*

107. Bullinger, H. and Seidal, U., 1989. *Assembly Sequence Planning Using Operational Networks.* in *10th ICPR.*

108. Bullinger, H. and Thaler, K., 1989. *Planning Of Flexible Assembly Based on Graphs - An Integrated Approach.* in *10th International Conference on Assembly Automation.*

109. Caselli, S. and Zanichelli, F., 1995. *On Assembly Sequence Planning Using Petri Nets.* in *Proceedings IEEE International Symposium on Assembly and Task Planning*: IEEE.

110. Chen, P., 1991. *Automatic Assembly Sequences Generation By Pattern Matching.* "IEEE Transactions on Systems, Man and Cybernetics", **21**(2): pp. 376-389.

111. Chen, C. and Pao, Y.-H., 1993. *An Integration of Neural Network and Rule-Based Systems for Design and Planning of Mechanical Assemblies.* "IEEE Transactions on Systems, Man, and Cybernetics", **23**(5): pp. 1359-1371.

112. Choi, C., Zha, X., Ng, T., and Lau, W., 1998. *On The Automatic Generation Of Product Assembly Sequences.* International Journal Of Production Research, **36**(3): pp. 617-633.

113. Barnes, C.J., Jared, G.E.M., and Swift, K.G., 2003. *A pragmatic approach to interactive assembly sequence evaluation.* Proceedings- Institution of Mechanical Engineers Part B Journal of Engineering Manufacture, **217**(4): pp. 541-550.

114. Tuttle, B., 1991. *Design For Function: A Cornerstone for DFMA.* in *International Forum on Product Design For Manufacture and Assembly.* Newport, RI (USA) - Boothroyd Dewhurst Inc.

115. QFD Institute, 2004, *Quality Function Deployment.* Online: http://www.qfdi.org/, Accessed on: 24.Feb.2004

116. Franceschini, F. and Rossetto, S., 1995. *QFD: The Problem of Comparing Technical/Engineering Design Requirements.* Research in Engineering Design, **7**: pp. 270-278.

117. Sthanusubramonian, T., Rinderle, J., and Finger, S., 1991. *Transformation from Functional Specification to Physical Form.* in *Proc. 1991 NSF Design and Manufacturing Systems Conf.*

118. Sthanusubramonian, T., Finger, S., and Rinderle, J., 1992. *A Transformational Approach to Configuration Design.* in *Design and manufacturing systems.* Atlanta; GA: Dearborn Mich.

119. Andersson, K., 1994. *A vocabulary for conceptual design.* in *Formal design methods for CAD.* Tallinn: Amsterdam.

120. Bashir, H.A. and Thomson, V., 2001. *Models for estimating design effort and time.* Design Studies, **22**(2): pp. 141-155.

121. Bashir, H.A. and Thomson, V., 2001. *An analogy-based model for estimating design effort.* Design Studies, **22**(2): pp. 157-167.

122. Sivaloganathan, S., Andrews, P.T.J., and Shahin, T.M.M., 2001. *Design function deployment: a tutorial introduction.* Journal of Engineering Design, **12**(1): pp. 59-74.

123. Sivaloganathan, S., Evbuomwan, N.F.O., Jebb, A., and Winn, H.P., 1995. *Design function deployment: a design system for the future.* Design Studies, **16**(4): pp. 447.

124. Mantripragada, R. and Whitney, D.E., 1998. *The Datum Flow Chain: A Systematic Approach to Assembly Design and Modeling.* Research in Engineering Design, **10**(3): pp. 150-165.

125. Lee, D.J. and Thornton, A.C., 1996. *The Identification and Use of Key Characteristics in the Product Development Process.* ASME Design Engineering Technical Conferences and Computers in Engineering Conference: pp. 309.

126. Thornton, A.C., 1999. *A Mathematical Framework for the Key Characteristic Process.* Research in Engineering Design, **11**(3): pp. 145-157.

127. Eppinger, S.D., Whitney, D.E., Smith, R.P., and Gebala, D.A., 1994. *A Model-Based Method for Organizing Tasks in Product Development.* Research in Engineering Design, **6**(1): pp. 1-13.

128. Yassine, A.A., Whitney, D.E., Lavine, J., and Zambito, T., 2000. *DETC2000/DTM-14547 Do-It-Right-First-Time (DRFT) Approach to Design Structure Matrix (DSM) Restructuring*. in *International conference on design theory and methodology*. Baltimore, MD: ASME.

129. Ishii, K., 1996. *Product Modularity: A Key Concept in Life-Cycle Design.* in *Frontiers of engineering.* Irvine; CA: National Academy Press.

130. Gershenson, J.K. and Prasad, G.J., 1998. *Life-Cycle Modularity in Product Design.* in *Vol 3; Design, product development and manufacturing.* Berlin: Society for Design and Process Science.

131. Zhang, Y. and Gershenson, J.K., 2003. *An Initial Study of Direct Relationships between Life-cycle Modularity and Life-cycle Cost.* Concurrent Engineering Research and Applications, **11**(2): pp. 121-128.

132. Marshall, R., Leaney, P.G., and Botterell, P., 1998. *Enhanced Product Realisation Through Modular Design: An Example of Product/Process Integration.* in *Vol 3; Design, product development and manufacturing.* Berlin: Society for Design and Process Science.

133. Ulrich, K.T. and Tung, K., 1991. *Fundamentals of Product Modularity.* in *Issues in Design/Manufacture Integration - 1991 American Society of Mechanical Engineers, Design Engineering Division (Publication).* New York, NY, USA: ASME.

134. Gershenson, J.K., Prasad, G.J., and Zhang, Y., 2003. *Product modularity: definitions and benefits.* Journal of Engineering Design, **14**(3): pp. 295-314.

135. Erixon, G., 1996. *Modular Function Deployment (MFD), support for good product structure creation.* in *Workshop on product structuring.* Delft, Netherlands: Delft University of Technology.

136. Erixon, G., 1996, *Design for Modularity*, in *Design for X: Concurrent Engineering Imperatives*, G.Q. Huang, Editor. Kluwer Academic Pub: London.
137. Sosa, M., Eppinger, S.D., and Rowles, C.M., 2000. *Designing modular and Interactive systems (DETC2000/DTM-14571)*. in *ASME 2000 Int'l Design Engineering Technical Conference and Computers and Information in Eng. Conference*. Baltimore, Maryland.
138. Van Wie, M.J., Greer, J.L., Campbell, M.I., Stone, R.B., and Wood, K.L., 2001. *Interfaces and product architecture (DETC01/DTM-21689)*. in *DETC'01 ASME Int'l Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Sept 9 - 12*. Pittsburgh, PA, USA: ASME.
139. Gu, P., Hashemian, M., Sosale, S., and Rivin, E., 1997. *An integrated modular design methodology for life-cycle engineering.* Annals- Cirp, **46**(1): pp. 71-74.
140. Holtta, K.M.M. and Otto, K.N., 2003. *Incorporating design complexity measures in architectural assessment (DETC2003/DTM-48648)*. in *Design Engineering Technical conference*. Chicago: ASME.
141. MacDuffie, J.P., Sethuraman, K., and Fisher, M.L., 1996. *Product Variety and Manufacturing Performance: Evidence from the International Automotive Assembly Plant Study.* Management Science, **42**(3): pp. 350-369.
142. Ulrich, K.T. and Eppinger, S.D., 1995, *Product design and development*. New York: McGraw-Hill. xxiii, 289 p.
143. Ishii, K., Juengel, C., and Fritz Eubanks, C., 1995. *Design for Product Variety: Key to Product Line Structuring*. in *Design theory and methodology*. Boston; MA: ASME.
144. Martin, M.V. and Ishii, K., 1996. *Design for Variety: A Methodology for Understanding the Costs of Product Proliferation.* ASME Design Engineering Technical Conferences and Computers in Engineering Conference: pp. 316.
145. Martin, M.V. and Ishii, K., 2002. *Design for variety: developing standardized and modularized product platform architectures.* Research in Engineering Design, **13**(4): pp. 213-235.
146. Martin, M.V. and Ishii, K., 1997. *Design for Variety: Development of Complexity Indices and Design Charts*. in *Proceedings ASME Design Engineering Technical Conference*.
147. Tang, V. and Salminen, V., 2001. *Towards a theory of complicatedness: Framework for complex systems analysis and design*. in *13th Int. Conference on Engineering Design*. Glasgow, Scotland.
148. Thilmany, J., 2003. *Too much information: Organizing information - after gathering it in the first place - is a key to actually using it.* Mechanical Engineering, (June): pp. 44 - 46.
149. Hauser, J.R., 2001. *Metrics thermostat.* Journal of Product Innovation Management, **18**: pp. 134-153.
150. Bribiesca, E., 1995. *Measuring 3D shape similarity using progressive transformations.* Pattern Recognition, **29**(7): pp. 1117-1129.
151. Wolpert, D.H. and Macready, W.G., 2000, *Self-Dissimilarity: An empirically observable complexity measure*, in *Unified Themes in Complex Systems:*

Proceedings of the First Necsi Int. Conf. on Complex Systems, Y. Bar-Yam, Editor.

152. Edmonds, B., 1997. *Complexity and scientific modelling*. in *20th International Wittgenstein Symposium*. Wechsel, Austria.
153. Hammond, R. and Bras, B.A., 1996. *Design for remanufacturing metrics*. in *1st Int'l workshop on reuse*. Eindhoven, The Netherlands (November 11-13).
154. Suri, G. and Luscher, A.F., 2000. *Evaluation metrics for the rating and optimization of snap-fits.* Research in Engineering Design, (12): pp. 191-203.
155. Shah, J.J., Vargas-Hernández, N., and Smith, S.M., 2002. *Metrics for measuring ideation effectiveness.* Design Studies, **24**(2): pp. 111-134.
156. Eckert, C., Clarkson, P.J., and Zanker, W., 2004. *Change and customisation in complex engineering domains.* Research in Engineering Design, **15**(1): pp. 1-21.
157. Kim, S.H. and Lee, K., 1989. *An Assembly Modelling System for Dynamic and Kinematic Analysis.* Computer-Aided Design, **21**(1): pp. 2-10.
158. Rajan, V., Lyons, K., and Sreerangam, R., 1997. *Assembly Representations For Capturing Mating Constraints and Component Kinematics*. in *Proceedings IEEE International Symposium on Assembly And Task Planning*: IEEE.
159. Anantha, R., Kramer, G., and Crawford, R., 1996. *Assembly Modelling by Geometric Constraint Satisfaction.* Computer Aided Design, **28**(9): pp. 707-722.
160. Uicker Jr., J.J., Pennock, G.R., and Shigley, J.E., 2003, *Theory of Machines and Mechanisms*. 3rd ed. ed. New York: Oxford University Press, Inc. 734.
161. Eppinger, S.D. and Salminen, V.K., 2001. *Patterns of product development interactions*. in *International conference on engineering design*. Glasgow: Bury St Edmunds.
162. Barnes, C. and Swift, K.G., 1999, *Chapter 11: Towards Assembly Sequence Generation In Design*, in *Computer Aided Process And Assembly Planning: Methods, Tools and Technologies.*
163. Zorc, S., Noe, D., and Kononenko, I., 1998. *Efficient Derivation Of The Optimal Assembly Sequence From Product Description.* Cybernetics and Systems, **29**(0): pp. 159-179.
164. Zha, X., Lim, S., and Fok, S., 1998. *Integrated Knowledge Based Assembly Sequence Planning.* International Journal Of Advanced Manufacturing Technology, **14**(0): pp. 50-64.
165. Rohrdanz, F., Mosemann, H., and Wahl, F., 1997. *Generating And Evaluating Stable Assembly Sequences.* Advanced Robotics, **11**(2): pp. 97-126.
166. Kaufman, S., Wilson, R., Jones, R., and Calton, T., 1996. *The Archimedes 2 Mechanical Assembly Planning System*. in *IEEE International Conference on Robotics and Automation.*
167. Wang, Y., Hseih, L., and Seliger, G., 1995. *Knowledge Based Integration Of Design and Assembly Process Planning*. in *Proceedings CIRP Seminars - Manufacturing Systems.*
168. Goldwasser, M., 1997, *Complexity Measures for Assembly Sequences*, Stanford University: Stanford. PhD Dissertation

169. Sodhi, R. and Turner, J., 1994. *Towards Modelling Of Assemblies For Product Design.* Computer Aided Design, **26**(2): pp. 85-97.

170. Shah, J., 1991. *Assessment Of Features Technology.* Computer Aided Design, **23**(5): pp. 331-343.

171. Karinthi, R.R. and Nau, D., 1992. *An Algebraic Approach to Feature Interactions.* IEEE Trans. Pattern Analysis and Machine Intelligence, **14**(4): pp. 469-484.

172. Woo, T., 1982. *Feature Extraction by Volume Decomposition.* in *"Proc. Conf. CAD/CAM Technology in Mechanical Engineering, Cambridge, USA".*

173. Rodriguez-Toro, C.A., Tate, S.J., Jared, G.E.M., and Swift, K.G., 2003. *Complexity metrics for design (simplicity + simplicity = complexity).* Proceedings- Institution of Mechanical Engineers Part B Journal of Engineering Manufacture, **217**(5): pp. 721-726.

174. Tate, S.J. and Jared, G.E.M., 2003. *Recognising symmetry in solid models.* Computer Aided Design, **35**(7): pp. 673-692.

175. Rodriguez-Toro, C.A., 2000, *Detection of n-fold Rotational Symmetry,* Cranfield University: Cranfield, Bedfordshire (UK). MSc Thesis

176. Corney, J.R. and Clark, D., 1993. *Face-based feature recognition: generalizing special cases.* Int. J. Computer Integrated Manufacturing, **6**(1 & 2): pp. 39 - 50.

177. Karling, A., 1949. *Group production and its influences on productivity.* in *2nd International Congress of Engineering Manufacture*. Paris.

178. Opitz, H., 1970, *A Classification System to Describe Workpieces*. Oxford: Pergamon Press.

179. Gallagher, C.C. and Knight, W.A., 1973, *Group Technology*: Butterworth.

180. Houtzeel, A., 1975. *MICLASS: A Classification System based on Group Technology.* in *SME Tech Pap MS75-721*. Los Angeles, CA (Mar 10-13).

181. Hyer, N.L. and Wemmerlov, U., 1989. *Group Technology in the US manufacturing industry: a survey of current practices.* International Journal Of Production Research, **27**(8): pp. 1287 - 1304.

182. Kaperthi, S. and Suresh, N.C., 1991. *A neural network system for shape-based classification and coding or rotational parts.* International Journal Of Production Research, **29**(9): pp. 1771 - 1784.

183. Nadir, Y., Chaabane, M., and Marty, c., 1993. *PROCODE - Automated coding system in group technology for rotational parts.* Computers in Industry, **23**: pp. 39 - 47.

184. Swift, K.G. and Booker, J., 2003, *Process Selection: From Design to Manufacture*. 2nd ed ed. London: Butterworh-Heinemann. 316 pp.

185. Whitney, D.E., 1993. *Nippondenso Co. Ltd: A Case Study of Strategic Product Design.* Research in Engineering Design, **5**(1): pp. 58.

186. Parry-Barwick, S. and Bowyer, A., 1993, *Feature Technology*. University of Bath: Bath, England. pp. 28. Technical Report. 001

187. Case, K. and Gao, J., 1993. *Feature Technology - An Overview*. International Journal Of Computer Integrated Manufacturing, **6**(1): pp. 2-12.

188. Regli, W.C., Gupta, S.K., and Nau, D.S., 1994, *Feature Recognition for Manufacturability Analysis*. University of Maryland. pp. 18. TR 94-10

189. Cicirello, V.A. and Regli, W.C., 2001. *Machining Feature-Based Comparison of Mechanical Parts*. in *SMI-2001: International Conference on Shape Modelling and Applications*. Genova, Italy.

190. Choi, B., Barash, M., and Anderson, D., 1984. *Automatic recognition of machined surfaces from a 3D solid model.* Computer Aided Design, **16**(2): pp. 81-86.

191. Joshi, S. and Chang, T., 1988. *Graph-Based Heuristics For Recognition Of Machined Features From A 3D Solid Model.* Computer Aided Design, **20**(2): pp. 58-66.

192. Kondo, M., 1994. *Decomposition Of Complex Geometry For A Manufacturing Application.* Computer Aided Design, **26**(3): pp. 244-252.

193. Gupta, S. and Nau, D., 1995. *Systematic Approach To Analysing The Manufacturability Of Machined Parts.* Computer Aided Design, **27**(5): pp. 323-342.

194. Han, J.-H., 1996, *Survey of Feature Research*. University of Southern California: Los Angeles, CA (USA). pp. 26. Technical Report. IRIS-96-346

195. Wu, M. and Liu, C., 1996. *Analysis On Machined Feature Recognition Techniques Based On B-Rep.* Computer Aided Design, **28**(8): pp. 603-616.

196. Bourne, D. and Wang, C.-H., 1995. *Design and Manufacture of Sheet Metal Parts: Using Features to Resolve Manufacturing Problems*. in *Proceedings of the ASME Computers in Engineering Conference and the Engineering Database Symposium*: ASME.

197. Kim, Y.S. and Wang, E., 2002. *Recognition of machining features for cast-then-machined parts.* Computer Aided Design, **34**: pp. 71-87.

198. Dissinger, T. and Magrab, E., 1996. *Geometric Reasoning For Manufacturability Evaluation - Application To Powder Metallurgy.* Computer Aided Design, **28**(10): pp. 783-794.

199. Kim, Y.S., Kim, Y., Pariente, F., and Wang, E., 1997. *Geometric Reasoning for Mill-turn machinig process planning.* Computers in Industial Engineering, **33**(3-4): pp. 501-504.

200. Senthil kumar, A., Salim, F.K., and Nee, A.Y.C., 1996. *Automatic Recognition of Design and Machining Features from Prismatic Parts.* Int. J. Advanced Manufacturing Technology, **11**(0): pp. 136-145.

201. Gupta, S.K., Regli, W.C., and Nau, D.S., 1994, *Manufacturing Features Instances: Which Ones to Recognize?* University of Maryland: Gaithersburg. pp. 18. TR 94-81

202. Gupta, S.K., Regli, W.C., Das, D., and Nau, D.S., 1997, *Automated Manufacturability Analysis: A Survey*. Carnegie Mellon University. pp. 36. NIST IR 5713

203. Swift, K.G., 1987, *Knowledge-based design for manufacture*. New technology modular series. London: Kluwer Academic Publishers.

204. Brissaud, D. and Tichkiewitch, S., 2000. *Innovation and manufacturability analysis in an integrated design context.* Computers in Industry, **43**: pp. 111 - 121.

205. Gupta, S.K., Regli, W.C., and Nau, D.S., 1994, *Integrating DFM with CAD through Design Critiquing*. Technical Report. TR 94-11

206. Liu, T., Yang, X., and Kalambur, G., 1995. *Design For Machining Using Expert System And Fuzzy Logic Approach.* Journal Of Materials Engineering Performance, **4**(5): pp. 599-609.
207. Sanchez, J., 1997. *Intelligent Reasoning Assistant For Incorporating Manufacturability Issues Into The Design Process.* Expert Systems With Applications, **12**(1): pp. 81-88.
208. Narang, R.V., 1997. *Recognition of neutral features for multiple domain analysis.* Computers and Industrial Engineering, **32**(3): pp. 539 - 543.
209. Jha, K. and Gurumoorthy, B., 2000. *Multiple feature interpretation across domains.* Computers in Industry, **42**: pp. 13- 32.
210. Subramani, S. and Gurumoorthy, B., 2003. *Associativity Between Feature Models Across Domains.* in *Proceedings Eighth ACM Symposium on Solid Modelling and Applications*. Seattle, Washington, USA.
211. Chen, Y.M., Wen, C.C., and Ted Ho, C., 2003. *Extraction of geometric characteristics for manufacturability assessment.* Robotics and Computer Integrated Manufacturing, **19**(4): pp. 371-385.
212. Miyakawa, S. and Ohashi, T., 1986. *The Hitachi Assemblability Evaluation Method.* in *Proceedings International Conference on Product Design For Assembly.*
213. Boothroyd, G., Dewhurst, P., and Knight, W., 1994, *Product Design for Manufacture and Assembly*. New York, USA: Marcel Dekker, Inc. 540.

Appendix A

MATRIX REPRESENTATION OF COMPONENT INTERACTIONS IN BINARY SPACE

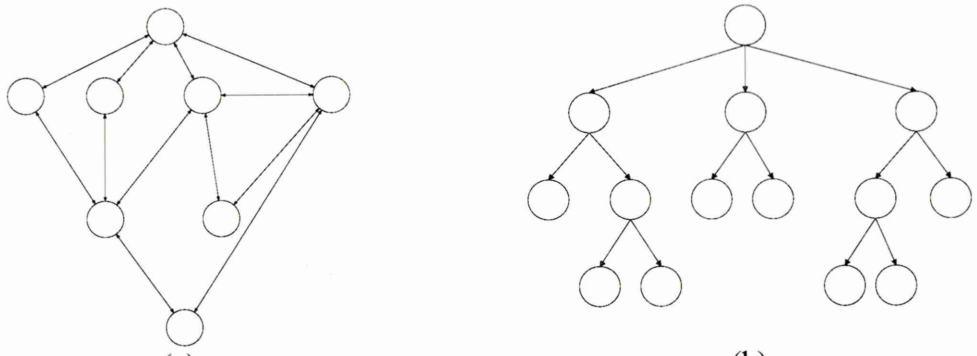
The architecture of a product lends itself to being symbolized in either of the two most well-known representation methods: graphs and matrices. There is abundant literature about graph algorithms (for instance, see Weiss[41]) and the convenience of its use to show several real-life problems, as well as sufficient information about matrix representation and manipulation.

12.1 Matrix representation

Graphs offer a powerful means of product architecture visualisation. A product can be represented in a graph by placing every component at a vertex, and the edge between any two vertices (components) would then represent their interaction. Moreover, these edges could be made to carry interaction information (relevance) by adding weights. Graphs also offer the possibility of only displaying the interactions that are actually there, which saves space in their representation, since non-existent interactions will not be represented.

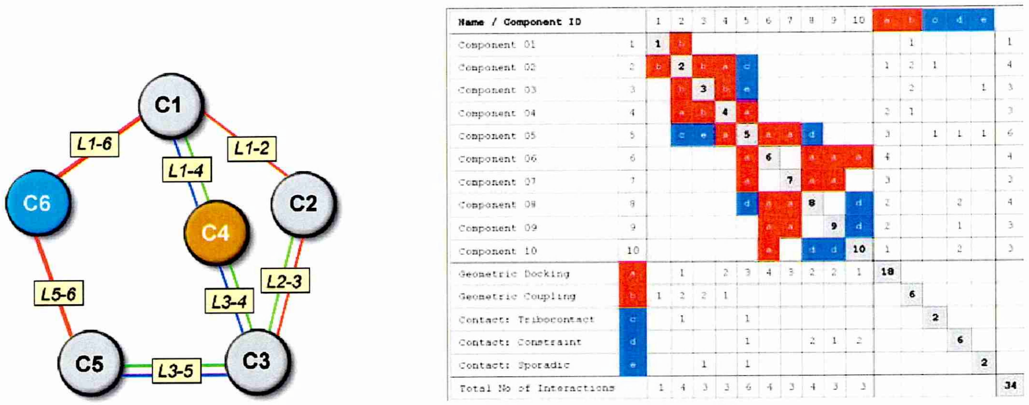
Furthermore, the various types of graphs available make it possible to represent product architectures based on part-to-part interactions, e.g. part networks or as a typical tree structure with branches expanding to represent subassemblies. (see Figure 12.1). However, despite these significant advantages offered by graphs, the preferred method of product representation, in this thesis for part-to-part interaction, is the *adjacency matrix representation*. (see Figure 12.2)

[41] Weiss, M.A., 1992. *Data structures and algorithm analysis*. Redwood City, Calif.: Benjamin/Cummings Pub. Co.; ISBN: 0805390529



(a) (b)

Figure 12.1 – Graphs representations (a) lattice and (b) trees



Name / Component ID		1	2	3	4	5	6	7	8	9	10	a	b	c	d	e	
Component 01	1	1	b										1				1
Component 02	2	b	2	b	a	c						1	2	1			4
Component 03	3		b	3	b	e							2			1	3
Component 04	4		a	b	4	a						2	1				3
Component 05	5		c	e	a	5	a	a	d			3		1	1	1	6
Component 06	6					a	6		a	a	a	4					4
Component 07	7					a		7	a	a		3					3
Component 08	8					d	a	a	8		d	2			2		4
Component 09	9						a	a		9	d	2			1		3
Component 10	10						a		d	d	10	1			2		3
Geometric Docking	a		1		2	3	4	3	2	2	1	18					
Geometric Coupling	b	1	2	2	1								6				
Contact: Tribocontact	c		1			1								2			
Contact: Constraint	d					1			2	1	2				6		
Contact: Sporadic	e			1		1										2	
Total No of Interactions		1	4	3	3	6	4	3	4	3	3						34

(a) (b)

Figure 12.2 – Part-to-part representation methods

The reason for selecting a matrix representation is due to its ease of use with any spreadsheet software package available in the market. It is also well-known that an adjacency matrix is another way to represent a graph using a two-dimensional array. For each part-to-part interaction, represented by an edge in a graph, there would be a numerical entry of 1; otherwise the entry in the array would be <u>void</u>. However, since every edge (interaction) is to be classified according to an assigned weighting protocol, the entry value in every cell that connects two components is this weight.

12.2 Adjacency matrix – entry method

At this point, it is not of particular interest to know whether the matrix is sparse or dense. Actually, the matrix can be represented only by the upper triangular part of it, where the diagonal is void, since any interactions between a component and itself are not represented.

The formation and data entry procedure of the metrics are described as follows:

1. For a set of components P = {C$_1$, C$_2$, C$_3$ ... C$_n$}, where *n* is the number of components available in a product. A matrix M is created such that all the components in P are placed in the leftmost column and similarly in the topmost row, as depicted on Figure 12.3-a.
2. The diagonal is left blank intentionally; otherwise it would imply comparing interactions of the i-th component with itself.
3. Starting with the first component in the leftmost column and travelling along this row, all interactions with every dissimilar component are accounted for, as depicted on Figure 12.3-b.
4. The value and type of entry to insert depends on the type of interface for that particular interaction. It can be any of the two geometrical types and/or any of the three adjacency types of interfaces as described in chapter 6.

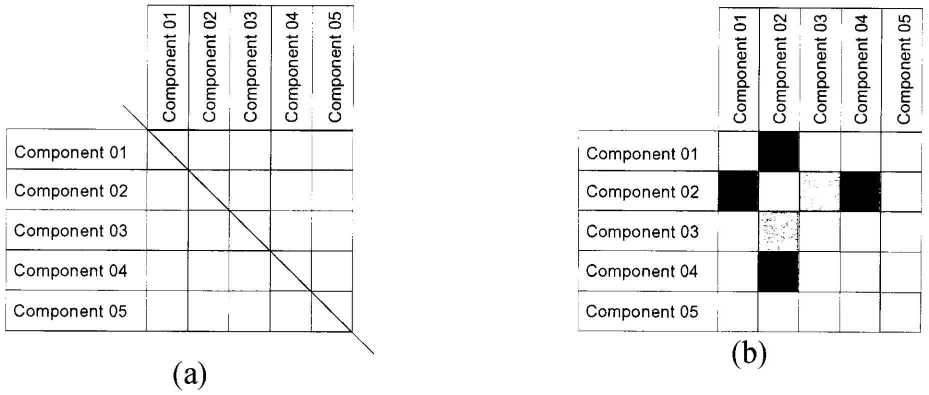


Figure 12.3 – How to fill in the adjacency matrix. (a) Matrix formation and (b) entries.

5. Weights are assigned to every interaction as if stacking up blocks. One interface can have more than one block, because an interface can have a particular geometric property and, additionally, a particular type of adjacency. Although, the representation remains in a two-dimensional space, these blocks are stacked up to represent the multi-dimensionality of complexity in an assembly. (See Figure 12.4). This procedure makes possible that, for instance, any two components, in the i-th row and the j-th column, can have an interaction weight of geometric value G1 plus adjacency value A1.
6. For every component in the i-th row, all the interactions need be accounted for. Counting interactions for every component will, inevitably, produce redundant information, i.e. interactions being counted twice, but it will generate information

for every component as seen later on. This information redundancy will create a symmetric matrix. In this procedure it is not important whether one component is placed before or after the next one –as in the planning of assembly sequences– the direction or incidence of the interactions is not taken into account.

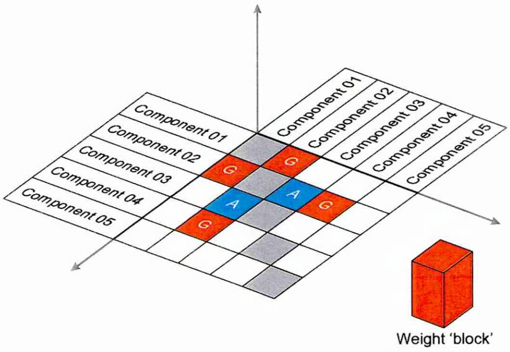
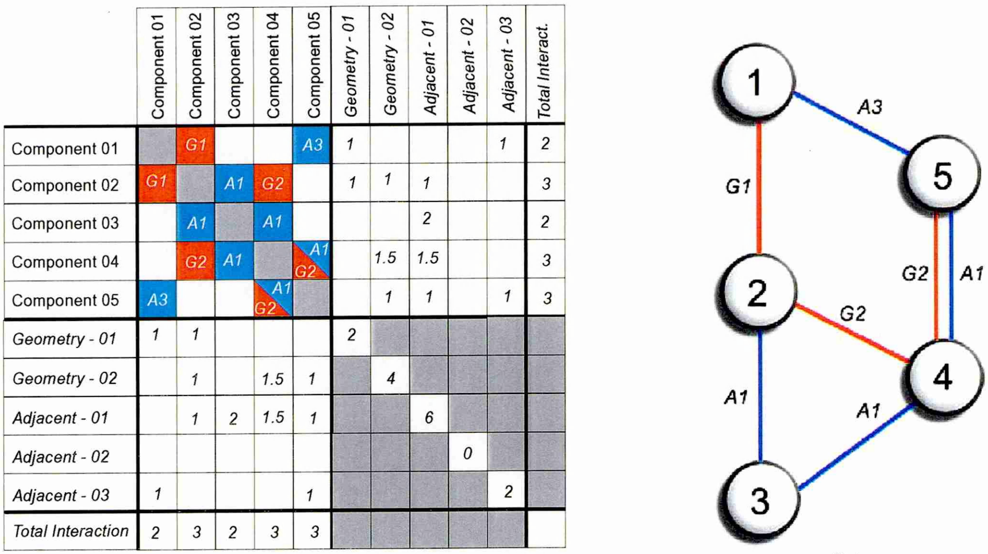


Figure 12.4 – Weighted interactions - Interface types

7. Once all the interactions, and therefore types of interfaces, have been assessed, the adjacency matrix can be extended to store the results. Figure 12.5(a) shows an extended matrix with groups of results equally displayed in the last six columns (on the right) as well as in the six rows (at the bottom.)



	Component 01	Component 02	Component 03	Component 04	Component 05	Geometry - 01	Geometry - 02	Adjacent - 01	Adjacent - 02	Adjacent - 03	Total Interact.
Component 01		G1			A3	1				1	2
Component 02	G1		A1	G2		1	1	1			3
Component 03		A1		A1				2			2
Component 04		G2	A1		A1 G2		1.5	1.5			3
Component 05	A3			A1 G2			1	1		1	3
Geometry - 01	1	1				2					
Geometry - 02		1		1.5	1		4				
Adjacent - 01		1	2	1.5	1			6			
Adjacent - 02									0		
Adjacent - 03	1				1					2	
Total Interaction	2	3	2	3	3						

(a) (b)

Figure 12.5 – (a) Adjacency matrix for any product architecure, (b) Equivalente on a graph representation

8. Following either group of results (last six columns or bottom six rows), every component has a summary of all types of interfaces that it contains. This method of discrimination is based on the idea of a component being nothing but a group

of interfaces, therefore its shape is dramatically influenced by two factors: (i) The task it is supposed to perform and (ii) the shape of neighbouring components.

9. Interactions that have two different types of interfaces (geometric + adjacent) are given a value of 0.5 for each type, such that every interaction is always counted as one.

10. Every component will display the total number of interactions with other components, this number will always be an integer, regardless of the types of interfaces that it has. For every component, there is also a count of interfaces per type, i.e. for the i-th component, there will be a count of the number of interfaces G1, G2, A1, A2 and A3. The number of the type of interfaces does not necessarily have to be an integer, for there might be cases (as explained above) where two types are present in the same interaction.

11. The total number of interactions per component is displayed at the bottom row (or last column), which can also be helpful to determine which component has the highest number of interactions.

12. Final the total of interactions within the product is equal to the sum of all the total number of component interactions, divided by 2 (to account for interactions)

13. The same result can be displayed in a graph if necessary (see Figure 12.5(b)).

Appendix B

SHAPE COMPLEXITY EVALUATION

Traditionally shape complexity has been evaluated based on the part classification according to its envelope and/or overall shape (see Figure 13.1).

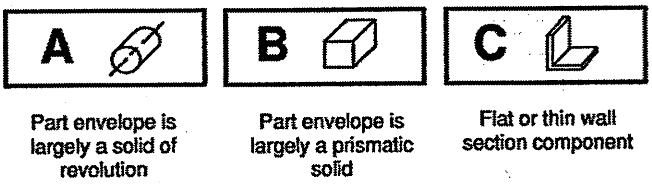
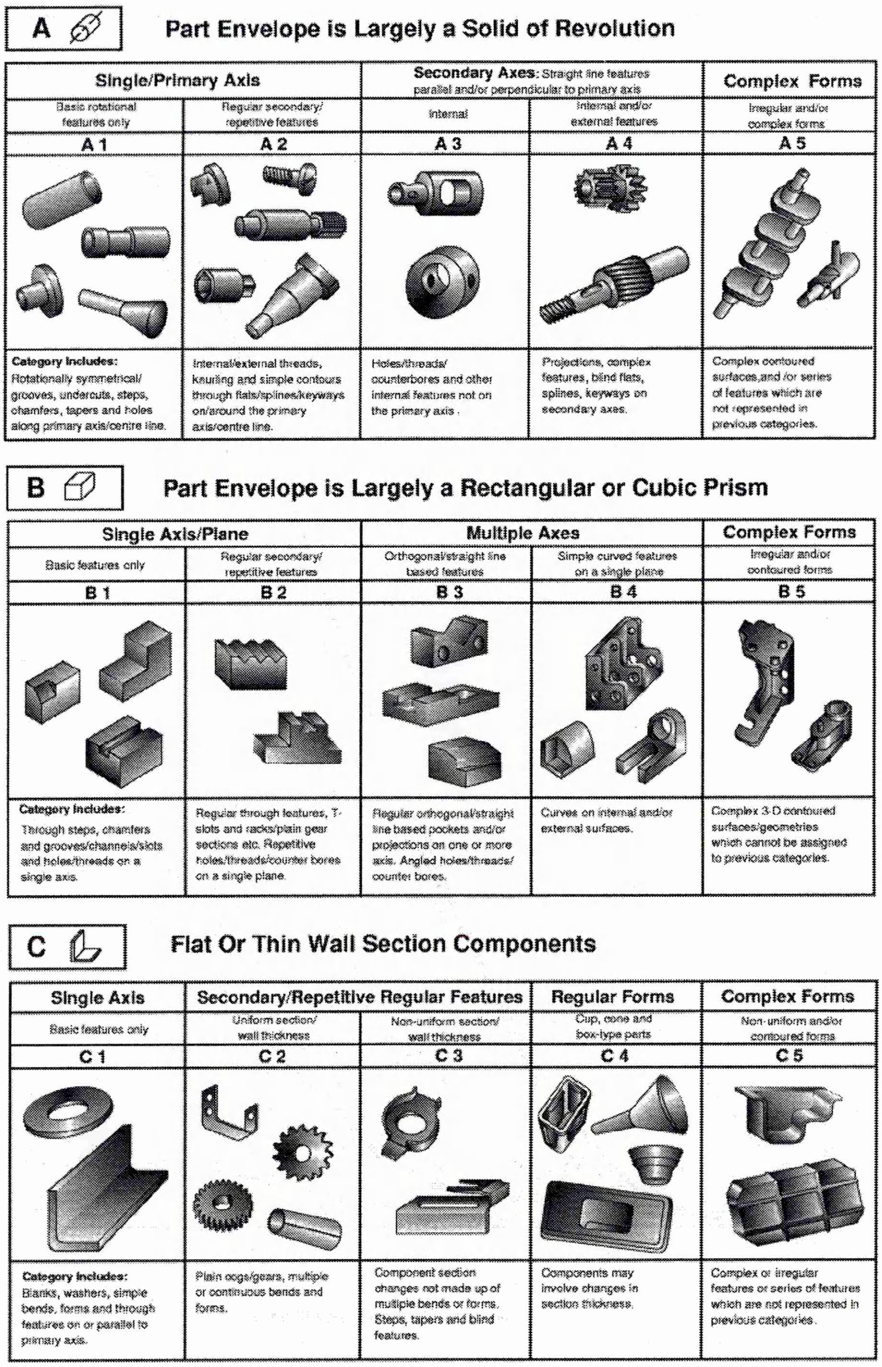

A
Part envelope is largely a solid of revolution
B
Part envelope is largely a prismatic solid
C
Flat or thin wall section component

Figure 13.1 – Part Envelope and wall categories

Once the part has been identified as A, B or C, the designer then inspected a chart that sub-classified every category in five additional types. (see Figure 13.2). Needless to say, these sub-divisions were entirely subjective and, therefore, prone to errors.



A Part Envelope is Largely a Solid of Revolution

Single/Primary Axis		Secondary Axes: Straight line features parallel and/or perpendicular to primary axis		Complex Forms
Basic rotational features only	Regular secondary/ repetitive features	Internal	Internal and/or external features	Irregular and/or complex forms
A 1	A 2	A 3	A 4	A 5
Category Includes: Rotationally symmetrical/ grooves, undercuts, steps, chamfers, tapers and holes along primary axis/centre line.	Internal/external threads, knurling and simple contours through flats/splines/keyways on/around the primary axis/centre line.	Holes/threads/ counterbores and other internal features not on the primary axis .	Projections, complex features, blind flats, splines, keyways on secondary axes.	Complex contoured surfaces,and /or series of features which are not represented in previous categories.

B Part Envelope is Largely a Rectangular or Cubic Prism

Single Axis/Plane		Multiple Axes		Complex Forms
Basic features only	Regular secondary/ repetitive features	Orthogonal/straight line based features	Simple curved features on a single plane	Irregular and/or contoured forms
B 1	B 2	B 3	B 4	B 5
Category Includes: Through steps, chamfers and grooves/channels/slots and holes/threads on a single axis.	Regular through features, T-slots and racks/plain gear sections etc. Repetitive holes/threads/counter bores on a single plane.	Regular orthogonal/straight line based pockets and/or projections on one or more axis. Angled holes/threads/ counter bores.	Curves on internal and/or external surfaces.	Complex 3-D contoured surfaces/geometries which cannot be assigned to previous categories.

C Flat Or Thin Wall Section Components

Single Axis	Secondary/Repetitive Regular Features		Regular Forms	Complex Forms
Basic features only	Uniform section/ wall thickness	Non-uniform section/ wall thickness	Cup, cone and box-type parts	Non-uniform and/or contoured forms
C 1	C 2	C 3	C 4	C 5
Category Includes: Blanks, washers, simple bends, forms and through features on or parallel to primary axis.	Plain cogs/gears, multiple or continuous bends and forms.	Component section changes not made up of multiple bends or forms. Steps, tapers and blind features.	Components may involve changes in section thickness.	Complex or irregular features or series of features which are not represented in previous categories.

Figure 13.2 – Shape classification categories