

**PRODUCT FAMILY DESIGN  
BASED ON A DESIGN REUSE MODEL**

**XU QIANLI**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2006**

**PRODUCT FAMILY DESIGN  
BASED ON A DESIGN REUSE MODEL**

**XU QIANLI**

(B.Eng., M.Eng., TJU)

**A THESIS SUBMITTED  
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY  
DEPARTMENT OF MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SINGAPORE**

**2006**

# ACKNOWLEDGEMENTS

I would like to thank my supervisors, Professor Andrew Nee Yeh Ching and Associate Professor Ong Soh Khim for their continual guidance, encouragement, and love throughout my graduate study in NUS. Their knowledge, insight and sincerity have been invaluable to my research, and will continue to be so in the years to come.

I would like to thank my Thesis Committee members for their comments and suggestions.

I give my special thanks to my parents and my brother, who have always been beside me with unreserved support, patience and love. They are the source of my hope and strength. Thanks also to Ms. Jiao Shunru for her patience, consideration and inspiration.

Thanks to my friends and colleagues for their support and discussions: Dr. Yuan Miaolong, Ms. Zhang Jie, Ms. Shen Yan, Mr. Louis Fong Wee Teck, Dr. Mani Mahesh, Mr. Cai Yanling and Mr. Chen Zhi. Many others have contributed to my research in various ways. Although their names were not mentioned here, I am obliged to all of them.

# SUMMARY

Product family design is a proven method to provide product variety while maintaining production efficiency. However, its application has been restricted by the lack of relevant information. Design reuse is a promising approach to alleviate this difficulty. However, current design reuse practices, such as case-based reasoning, catalog-based design and modular design, have only focused on one or a few aspects of product family design. A complete design reuse process model has not been defined. Therefore, this research aims to develop the design reuse methodology to support product family design.

A product family design reuse (PFDR) process model was developed to accommodate the major issues of product family design. This model incorporates information modeling, information processing, and design synthesis and evaluation into a holistic model. Thus, it provides systematic support to build product platforms and design product families.

A multiple facet information model was developed to decompose existing product cases. It can deal with heterogeneous product information with sufficient flexibility and representation rigor. A function-based product architecture was established with the assistance of a new analytical tool, namely, the self-organizing map (SOM). Based

on a formal presentation of the product functions, the SOM can cluster the product functions without human supervision. In comparison to traditional methods that depend on manual operations or heuristic rules, the SOM method is fast and relies less on human intelligence. The SOM method, in combination with a few other knowledge extraction operations, enables a more efficient reuse of the product information.

Product performance was evaluated using the information content, which incorporates diverse measures of product performance criteria into a dimensionless metric. The information content assessment (ICA) method defines logic procedures to establish the system ranges of components, and compute the information content. This is an improvement to the previous methods where the information content was computed subjectively. Information content is used as an objective function in product family design and optimization, through which product performance can be better predicted.

The PFDR methodology has been used in three product family design tasks. The design of cellular phone products shows the effectiveness of PFDR in automated design synthesis and evaluation. The design of TV receiver circuits demonstrates the advantages of the design reuse method as compared to the modular design method. In the case of the fan filter unit (FFU) design, the design reuse method was benchmarked against the traditional experience-based method. It was shown that the PFDR method can achieve a more efficient product family design with respect to product quality and cost.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS .....</b>	<b>i</b>
<b>SUMMARY .....</b>	<b>ii</b>
<b>TABLE OF CONTENTS .....</b>	<b>iv</b>
<b>LIST OF FIGURES .....</b>	<b>viii</b>
<b>LIST OF TABLES.....</b>	<b>xi</b>
<b>NOMENCLATURE .....</b>	<b>xiii</b>
<b>Chapter 1 INTRODUCTION.....</b>	<b>1</b>
<b>1.1 Product Conceptual Design.....</b>	<b>4</b>
1.1.1 Conceptual design .....	4
1.1.2 Product family design .....	5
<b>1.2 Engineering Design Reuse .....</b>	<b>7</b>
1.2.1 Types of design reuse .....	8
1.2.2 Design reuse processes.....	9
1.2.3 Product information modeling and analysis.....	10
1.2.4 Design synthesis and evaluation .....	12
<b>1.3 Research Objectives .....</b>	<b>13</b>
<b>1.4 Thesis Structure.....</b>	<b>15</b>
<b>Chapter 2 LITERATURE REVIEW .....</b>	<b>17</b>
<b>2.1 Fundamentals Of Product Family Design.....</b>	<b>17</b>
2.1.1 Top-down approaches .....	17
2.1.2 Bottom-up approaches .....	20

<b>2.2</b>	<b>Design Reuse For Product Family Design.....</b>	<b>21</b>
2.2.1	Representation of product information .....	22
2.2.2	Establishment of product architecture.....	25
2.2.3	Product family design as a configuration design problem.....	28
2.2.4	Optimization and solution evaluation .....	30
2.2.5	Look back and look ahead.....	33
<b>2.3</b>	<b>Summary .....</b>	<b>39</b>
<b>Chapter 3</b>	<b>FRAMEWORK OF PRODUCT FAMILY DESIGN REUSE.....</b>	<b>41</b>
<b>3.1</b>	<b>Integrated Design Reuse Process Model .....</b>	<b>41</b>
3.1.1	Stage I: Product information modeling .....	42
3.1.2	Stage II: Knowledge extraction.....	43
3.1.3	Stage III: Design synthesis and evaluation .....	46
<b>3.2</b>	<b>Prerequisites And Problem Boundaries .....</b>	<b>47</b>
3.2.1	Prerequisites .....	47
3.2.2	Problem boundaries.....	48
<b>Chapter 4</b>	<b>ESTABLISHMENT OF PRODUCT PLATFORM.....</b>	<b>50</b>
<b>4.1</b>	<b>Function-Based Product Information Model .....</b>	<b>50</b>
4.1.1	Product information representation.....	50
4.1.2	The key element vector representation of function structure.....	54
4.1.3	Function and flow taxonomies .....	56
<b>4.2</b>	<b>Building Of FPA Using Self-Organizing Map.....</b>	<b>60</b>
4.2.1	Introduction of SOM .....	62
4.2.2	Function clustering based on SOM .....	64
4.2.3	An illustrative example .....	69
4.2.4	Evaluation of the SOM method.....	75
<b>4.3</b>	<b>Establishment Of Product Platform .....</b>	<b>77</b>
4.3.1	Extraction of KCs as performance criteria.....	79

4.3.2	Formation of component catalog .....	79
4.3.3	Establishment of mapping route using correlation matrices .....	80
<b>4.4</b>	<b>SUMMARY.....</b>	<b>84</b>
<b>Chapter 5</b>	<b>ICA METHOD FOR PRODUCT PERFORMANCE EVALUATION</b> .....	<b>85</b>
<b>5.1</b>	<b>Product Performance Evaluation .....</b>	<b>85</b>
<b>5.2</b>	<b>The Information Content Assessment (ICA) Method.....</b>	<b>86</b>
5.2.1	Background .....	86
5.2.2	Procedures of the ICA method .....	89
5.2.3	Establishment of system range from existing products.....	90
5.2.4	Calculation of information content .....	97
5.2.5	A comparison of the ICA method and axiomatic design.....	100
<b>5.3</b>	<b>Precautions And Limitations.....</b>	<b>102</b>
<b>5.4</b>	<b>Summary .....</b>	<b>104</b>
<b>Chapter 6</b>	<b>MULTIPLE OBJECTIVE OPTIMIZATION FOR DESIGN</b> <b>SYNTHESIS .....</b>	<b>105</b>
<b>6.1</b>	<b>Problem Formulation.....</b>	<b>105</b>
<b>6.2</b>	<b>Establishment Of Product Family Cost Model.....</b>	<b>108</b>
6.2.1	Cost structure and cost model .....	108
6.2.2	An empirical cost model for product family design.....	110
<b>6.3</b>	<b>Multiple Objective Optimization .....</b>	<b>114</b>
6.3.1	Introduction of multiple objective optimization problem .....	115
6.3.2	Multi-objective struggle genetic algorithm .....	117
6.3.3	Important issues in the optimization algorithm.....	119
<b>6.4</b>	<b>Post-Optimal Solution Selection .....</b>	<b>124</b>



<b>6.5</b>	<b>Summary .....</b>	<b>126</b>
<b>Chapter 7</b>	<b>SYSTEM IMPLEMENTATION AND CASE STUDIES .....</b>	<b>127</b>
<b>7.1</b>	<b>A Prototype Product Family Design Reuse System.....</b>	<b>127</b>
<b>7.2</b>	<b>Case Study I: Cellular Phone Product Family Design .....</b>	<b>131</b>
7.2.1	Settings.....	132
7.2.2	Results .....	136
7.2.3	Discussion .....	138
<b>7.3</b>	<b>Case Study II: TV Receiver Circuits Design.....</b>	<b>139</b>
7.3.1	Settings.....	140
7.3.2	Solution generation and results .....	142
7.3.3	Discussion .....	144
<b>7.4</b>	<b>Case Study III: Fan Filter Unit Design.....</b>	<b>147</b>
7.4.1	Establishment of product platform.....	148
7.4.2	Configuration design of FFU using two methods .....	152
7.4.3	Discussion .....	162
<b>7.5</b>	<b>Summary .....</b>	<b>164</b>
<b>Chapter 8</b>	<b>CONCLUSIONS AND FUTURE WORK.....</b>	<b>165</b>
<b>8.1</b>	<b>Conclusions .....</b>	<b>165</b>
<b>8.2</b>	<b>Future Work .....</b>	<b>169</b>
	<b>PUBLICATIONS FROM THIS THESIS .....</b>	<b>172</b>
	<b>REFERENCES.....</b>	<b>173</b>
	<b>APPENDICES .....</b>	<b>187</b>
<b>APPENDIX A</b>	<b>FLOW TAXONOMY.....</b>	<b>187</b>
<b>APPENDIX B</b>	<b>FUNCTION TAXONOMY .....</b>	<b>188</b>

# LIST OF FIGURES

Figure 1.1	Current and foreseeable benefits of design reuse (Duffy and Ferns, 1999) .....	4
Figure 1.2	A product development road-map .....	6
Figure 1.3	A design reuse process model (Duffy <i>et al.</i> , 1995) .....	10
Figure 2.1	A process of top-down product family design .....	18
Figure 2.2	A process of bottom-up product family design .....	21
Figure 3.1	The PFDR process model.....	42
Figure 4.1	Data structure of function and flow.....	51
Figure 4.2	Data structure of KCs.....	52
Figure 4.3	Data structure of physical components .....	53
Figure 4.4	Data structure of contextual information .....	53
Figure 4.5	A block representation of function - ‘heat generation’ .....	55
Figure 4.6	An excerpt of function action and flow taxonomies .....	59
Figure 4.7	Coding schemes of function action and flow taxonomies .....	59
Figure 4.8	Self-organizing map: the Kohonen model (Haykin, 1999).....	63
Figure 4.9	Graphical interpretation of function clustering .....	65
Figure 4.10	Neighborhood activation in a hexagonal lattice.....	67
Figure 4.11	Updating weight vector in a 2D plane.....	68
Figure 4.12	Function structure of an electric kettle.....	70

Figure 4.13	Initial status of the competitive layer .....	73
Figure 4.14	Clustering pattern in the competitive layer after training .....	73
Figure 4.15	FPA of the electric kettle products .....	75
Figure 4.16	Mapping route from design requirements to design parameters.....	78
Figure 4.17	Mapping route from CRs to physical components.....	82
Figure 5.1	Relationship between design range and system range .....	87
Figure 5.2	The processes of the ICA method .....	90
Figure 5.3	Computing the component capability indices .....	92
Figure 5.4	Capability index for component combination( $m_2^1, m_4^1$ ) .....	95
Figure 5.5	Typical capability indices for different types of KCs.....	95
Figure 6.1	Problem formulation of design synthesis and evaluation .....	107
Figure 6.2	Cost model of a product family.....	111
Figure 6.3	Cost road-maps of cellular phone batteries .....	112
Figure 6.4	Flowchart of MOSGA for the design synthesis problem.....	117
Figure 6.5	Visualization of population energy convergence .....	123
Figure 6.6	Visualization of population energy and population plot .....	123
Figure 6.7	Pareto-front and post-optimal solution selection .....	125
Figure 7.1	Architecture of the PFDR prototype system .....	128
Figure 7.2	User interface for product information modeling .....	129
Figure 7.3	User interface for product function decomposition.....	129
Figure 7.4	User interface for design synthesis.....	131
Figure 7.5	Objective functions of the solutions w.r.t. different priority strategies	137

Figure 7.6	Component capability indices of three deflection circuit components	142
Figure 7.7	FFU structure and major components .....	147
Figure 7.8	Function structure of FFU .....	149
Figure 7.9	Feature map in the competitive layer (FFU) .....	150
Figure 7.10	FPA of FFU products.....	150
Figure 7.11	Motors used in $P_A$ and $P_B$ .....	155
Figure 7.12	Standard casing structure ( $P_B$ ).....	156
Figure 7.13	Redesigned casing structure ( $P_A$ ) .....	157

# LIST OF TABLES

Table 2.1	Summary of product family design approaches.....	35
Table 4.1	Input vector of an atomic function – ‘heat generation’ .....	66
Table 4.2	Atomic functions of four sample products.....	71
Table 4.3	Normalized input data of the atomic functions .....	72
Table 4.4	Correlation between CRs and KCs ( $TR_1$ ) .....	81
Table 4.5	Correlation between KCs and functions ( $TR_2$ ) .....	82
Table 5.1	KCs of the electric kettle.....	91
Table 5.2	Correlation between KCs and functions ( $TR_2$ ) .....	91
Table 5.3	Function and component slot .....	91
Table 5.4	Sampled <i>power consumption</i> values of three products that host ( $m_2^1, m_4^1$ ) .....	94
Table 5.5	Design requirements of a family of electric kettle products .....	97
Table 5.6	Product configurations of an electric kettle.....	99
Table 5.7	Computation of information content.....	99
Table 6.1	A few representative cost models.....	109
Table 6.2	Chromosome structure of the electric kettle product family ( $N=3$ ).....	120
Table 7.1	KCs of the cellular phones .....	132
Table 7.2	Correlation between KCs and functions ( $TR_2$ ) .....	133
Table 7.3	Function and component slot .....	133

Table 7.4	Sampled <i>MTT</i> values of five products that host (RFC-T1, Li-Ion 860).....	135
Table 7.5	Design requirements of the product family.....	136
Table 7.6	Product configurations (performance priority) .....	136
Table 7.7	Product configurations (equal priority).....	137
Table 7.8	Product configurations (cost priority).....	137
Table 7.9	Product variety of TV sets.....	140
Table 7.10	Mapping from design requirements to components.....	141
Table 7.11	Cost model reformulation based on Fujita <i>et al.</i> (1999) .....	143
Table 7.12	Optimization results of product family cost.....	144
Table 7.13	Comparison of the PFDR method and the benchmark method.....	146
Table 7.14	KCs of the FFU .....	151
Table 7.15	Correlation between KCs and functions ( $TR_2$ ) .....	151
Table 7.16	Function and component slot .....	151
Table 7.17	Design requirements of the FFU product.....	152
Table 7.18	$P_A$ – product configuration generated by the experience-based method....	153
Table 7.19	$P_B$ – product configuration generated by the PFDR method.....	154
Table 7.20	Performance of $P_A$ .....	158
Table 7.21	Computation of information content.....	160
Table 7.22	Performance of $P_B$ .....	162

# NOMENCLATURE

2D	Two dimensional
$A_F$	A function action
AI	Artificial intelligence
$C(\mathbf{m})$	Cost of a product family
CAD	Computer-Aided Design
$C_F$	Fixed cost
$C_c^j$	Cost of component $j$
$C_d^i$	Development cost of product $i$
$C_p^i$	Cost of product $i$
CML	Compositional modeling language
CR	Customer requirement
$D_{IST}$	Combined distance
$D_{IST}^S$	Distance in the parameter space
$D_{IST}^T$	Distance in the attribute space
DOE	Design of experiments
$E_{NG}$	Flow of energy
$\mathbf{f}$	A vector of common functions of a product family
$\mathbf{F}_i$	Key element vector representation of the function structure of $p_i$
$f_i$	A common function of a product family

$F_i^0$	Function structure of a product $p_i$
$\vec{f}_i$	An atomic function represented as a key element vector
FPA	Function-based product architecture
FR	Functional requirement
GA	Genetic algorithm
GUI	Graphical User Interface
$\mathbf{h}$	A vector of host products
ICA	Information content assessment
$I$	A scalar of information content
$I_W$	Input flow
$\mathbf{k}$	A vector of key characteristics of a product family
$k_i$	A key characteristic
$K_i^0$	A vector of key characteristics of a product $p_i$
KC	Key characteristic
KEV	Key element vector
$M_i^0$	Vector of physical components of a product $p_i$
$\mathbf{m}$	A vector of physical modules in the component catalog
$M_{AT}$	Flow of material
MOSGA	Multi-objective struggle genetic algorithm
MTBF	Mean time between failures
OEM	Original equipment manufacturer
$O_W$	Output flow



$Op_c$	Knowledge extraction operator – cost modeling
$Op_f$	Knowledge extraction operator – function analysis
$Op_i$	Knowledge extraction operator – component capability index
$Op_k$	Knowledge extraction operator – KC extraction
$Op_r$	Knowledge extraction operator – correlation matrix
$\mathbf{P}$	A product family to be designed
PFDR	Product family design reuse
$P_i$	A product to be designed
$p_i$	An existing product case
pdf	Probability density function
pmf	Probability mass function
QFD	Quality Function Deployment
$\mathbf{r}$	A vector of customer requirements
RSM	Response surface method
$ S $	Feasible design space
$s_i$	A component slot
$S_{IN}$	Flow of signal
SA	Simulated annealing
SOM	Self-organizing map
$SP_i$	A general design space
STEP	Standard for the exchange of product model data
$T(\mathbf{m})$	Objective function (fitness function)

$T_w(\mathbf{m})$	Weighted fitness of a solution
$TR_1$	Correlation matrix between CRs and KCs
$TR_2$	Correlation matrix between KCs and functions
$TR_3$	Correlation matrix between functions and physical components
UML	Unified modeling language
$\bar{w}_i$	A vector of weight
$w_i$	A scalar value of weight
$X_i^0$	Contextual data of a product $p_i$
XML	Extensible markup language
$\alpha^i$	Cost coefficient of complexity
$\mathbf{v}$	A vector of component attributes
$\kappa$	A key element
$\mu$	A scalar value of mean
$\sigma$	A scalar value of standard deviation
$\zeta$	A scalar value of probability

**To my parents**

# Chapter 1 INTRODUCTION

Today's market is characterized by intense competition in the global manufacturing environment. In order to succeed or even to survive, a manufacturer must be able to deliver their products with speed, diversity, high quality, and at low cost. Product design is the key factor to meet these requirements. Among the several stages of product design, which usually encompass requirement analysis, conceptual design, embodiment design, and detailed design, the conceptual design stage is of paramount importance. This can be shown with two observations. Firstly, the conceptual stage allows for the maximum design freedom, i.e., the designer is less constrained to make decisions at this stage. Secondly, the cost of a product is largely determined at this stage. It is estimated that about 75% of the manufacturing cost is committed by the end of the conceptual stage (Ullman, 1997). In the subsequent stages, it becomes increasingly difficult and costly to compensate for the initial flawed designs.

In conceptual design, the target can be designing a single product or a set of related products, i.e., a product family. Product family design is a nascent but rapidly maturing field of research (Simpson, 2004). The fundamental idea is to address diverse customer requirements with a product family, while maintaining economies of scale of production. However, in such an effort, one difficulty is significant, namely, a lack of information. In fact, the early design stage is characterized by information deficiency

and uncertainty (Simpson *et al.*, 1998; Wood and Agogino, 2004). Thus, there is apparently a paradox: when the maximum value of a product is determined, minimal information is available to support it.

Design reuse provides a possible means to address this difficulty. Systematic design reuse methodologies can be applied to facilitate product family design at the conceptual stage. To do so, three fundamental questions have to be answered.

- (1) Why is design reuse necessary?
- (2) Is it possible to apply design reuse?
- (3) Is the design reuse methodology effective in product design?

**Necessity** – It makes little sense to reinvent the wheel. In today’s market, no enterprise can afford the time and resources to design an entire product from scratch. Reuse of prior knowledge is crucial to design rapidity and continuity. Effective product design requires an efficient retrieval and utilization of information. However, designers are constantly frustrated by the lack of means to access the relevant information. This is not necessarily caused by the paucity of product data. Instead, the proliferation of data makes the retrieval of relevant information a daunting task. Therefore, the designer is in a dilemma of being “drown in data but thirsty for knowledge” (Rezayat, 2000). There is an urgent need for effective information management based on design reuse.

**Applicability** – In order to apply design reuse, it is required that a set of designed products already exist and the related design information is accessible. This should not be a problem for an established company because there is usually a pool of designed products. Typical in the industry, product development is evolutionary rather than revolutionary. According to statistics, only about 20% of an OEM's investment is on new design while about 80% is on the reuse of existing products, with or without modification (Rezayat, 2000). Thus, design reuse can be applied in a broad variety of industries. The question is: how to organize the information such that reuse is technically feasible and cost effective.

**Effectiveness** – The effectiveness of design reuse should be validated by the improvements in the key factors of production, namely, cost, quality, and time-to-market. It is expected that production efficiency can be increased because the designers do not have to start from scratch. Product quality can be improved by reusing the sub-systems or components which quality and validity have been proven (Li *et al.*, 2004). In addition, the outcome of the design can be better predicted, which is valuable to the early decision-making stage. By properly reusing existing technologies, significant benefits can be achieved with respect to cost, time, product quality and performance (Duffy and Ferns, 1999) (Figure 1.1).

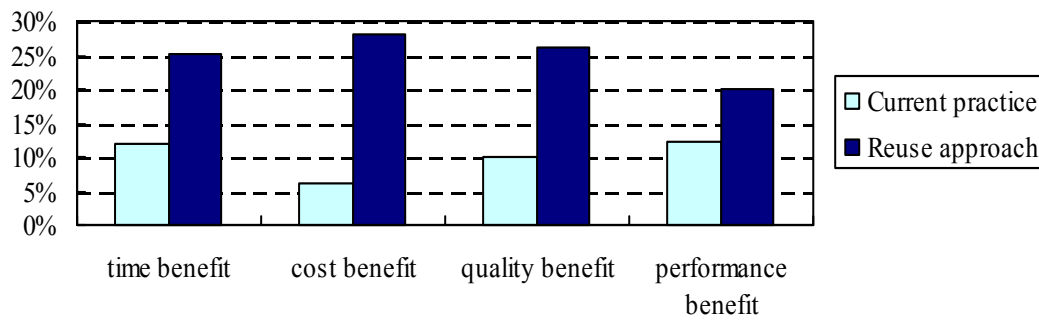


Figure 1.1 Current and foreseeable benefits of design reuse (Duffy and Ferns, 1999)

To support design reuse activities, it is necessary to understand the characteristics of conceptual design and product family design. It is also important to be aware of the capabilities of design reuse and the available tools and techniques. These topics are discussed in Sections 1.1 and 1.2, respectively.

## 1.1 Product Conceptual Design

In this research, conceptual design refers to the activities that determine the schematic principles and structures of a product that lead to the desired functionalities. The major research issues are presented next.

### 1.1.1 Conceptual design

Conceptual design is a design process that involves intense decision-making. A systematic, procedural process model must be developed to manage these decision-making activities. A few notable design theories that have dealt with this problem include the systematic approach (Pahl and Beitz, 1996), total design (Pugh,

1991), robust design (Clausing, 1994), the mechanical design process (Ullman, 1997), axiomatic design (Suh, 2001), etc.

At the early design stages, decisions have to be made on the project definition, design specifications, concept generation, concept evaluation, and the preliminary production issues. The effectiveness in carrying out these activities depends a lot on the availability of information, and the way in which the information is processed. Since the conceptual design stage is characterized by information deficiency and uncertainty, a paramount problem is how to carry out design based on the limited amount of information. Collection of information from existing products is a possible way to solve the problem. However, product information is highly unstructured and appears in diverse forms. Significant effort is required to represent and capture product information, and utilize the information in new design problems.

### **1.1.2 Product family design**

Product family refers to a group of related products that share common technologies and address a series of market segmentations (Meyer and Lehnerd, 1997). The rationale of product family design is to provide product variety while maintaining production efficiency (Pine, 1993). Product variety is defined in terms of customer requirements, which are addressed by variegated product performance. Thus, a product family has to be designed to cover a ranged set of performance requirements. At the same time, production efficiency has to be ensured by considering commonality,



compatibility, standardization and modularity among different products (Meyer and Lehnerd, 1997). This is achieved through developing common technologies and components, which can be shared among different products. In practice, a product development road-map is often designed to manage the evolution of products in a corporation. As shown in Figure 1.2, the horizontal axis is the time divided into years and quarters. The products (denoted as hexagons) are distributed in three tiers, namely, the high tier, mid tier and mass tier, according to the market segmentations shown on the vertical axis. The curve on the right shows the production volume in the different market segmentations. From the road-map, it can be observed that there is a constant migration of technologies from the higher end to the lower end as time proceeds. This ensures the continuation of product development within a corporation.

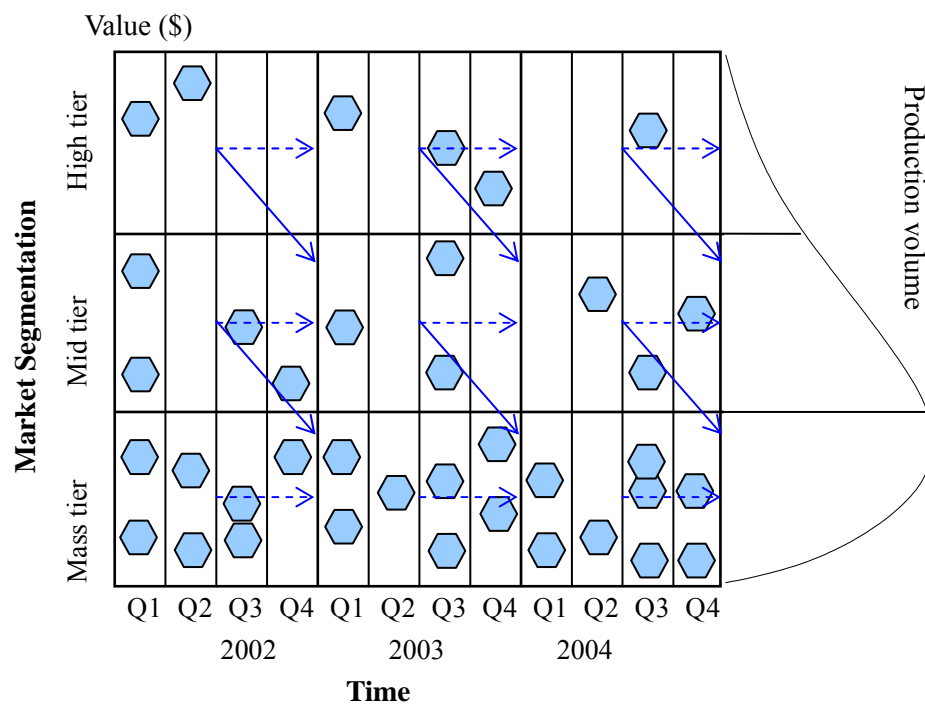


Figure 1.2 A product development road-map

The major concern in product family design is the management of the trade-offs between product commonality and product performance. Usually, increased commonality leads to higher production efficiency; but at the expense of product performance. Decisions have to be made at the early design stages about (1) the proper divisions of market segmentations, (2) the structure and content of a product platform, (3) the attributes of the common components under the product platform, and (4) the optimal combination and adaptation of components. Thereafter, it is also important to evaluate (5) the effectiveness of the product family with respect to cost and product performance.

Information deficiency and uncertainty is a big hindrance to product family design. Usually, a designer is faced with immense freedom to develop the product family. It is not trivial to set the right parameters as a good starting point, e.g., little is known about the consequences of setting a parameter at a specific value. Therefore, it is necessary to find ways to collect the relevant information and use it to ensure design optimality.

## **1.2 Engineering Design Reuse**

Design reuse involves various activities that utilize existing technologies to address new design problems. Different forms of design reuse are discussed in Section 1.2.1. Design reuse activities must be carried out according to proper procedures. Thus, the management of the design process becomes imperative (Section 1.2.2). A few major

issues, namely, information modeling and analysis, and design synthesis and evaluation are discussed in Section 1.2.3.

### **1.2.1 Types of design reuse**

Basically, reuse is divided into three forms with respect to the objects to be reused.

- (1) End-of-life product reuse, which refers to the reuse and recycling of obsolete products or components such that the components or materials can return to the product life cycle. This results in savings of natural resources and reduction of environmental impacts (Hata *et al.*, 1997; Kimura *et al.*, 1998).
  
- (2) Reuse of existing manufacturing resources. The manufacturing process inevitably consumes energy and resources, especially when the manufacturing equipments have to be redesigned, upgraded, or reconfigured. Production cost can be reduced through the utilization of existing manufacturing resources to accommodate the changing production requirements (Kimura and Nielsen, 2005).
  
- (3) Reuse of product information and design knowledge. This type of reuse is a pre-requisite of the other two types of reuse because design ultimately determines the extent to which the products and the manufacturing resources can be reused. In other words, effective reuse of available resources could not be achieved unless the products are designed to be reusable.

This research focuses on the third type of design reuse, i.e., *the various approaches that support the utilization of knowledge gained from previous design activities*. This is based on the belief that knowledge/information reuse enables the reuse of components and manufacturing resources, and hence is essential to sustainable design and manufacturing.

### **1.2.2 Design reuse processes**

Systematic design reuse method involves two interrelated processes: information collection and information reuse. The former refers to design-for-reuse, which involves information modeling and information processing to identify relevant knowledge. The latter refers to design-by-reuse, which aims at the effective utilization of the information. Design-by-reuse is mainly concerned with information retrieval, solution synthesis and evaluation.

To properly organize the design reuse process, a comprehensive design reuse process model is required. Various methods have been developed, such as case-based reasoning (Watson, 1999; William and Agogino, 1996), catalog-based design (Chakrabarti and Bligh, 1996), modular design (Fujita *et al.*, 1999; McAdams *et al.*, 1999), etc. These methods, however, have been criticized for depending on non-holistic models, i.e., the overall design process has not been well-organized (Smith, 2002). A relatively complete design reuse process model was proposed by Duffy *et al.* (1995). It consists of three processes and six knowledge resources (Figure 1.3). An effective design reuse

system has to provide tools to facilitate the design processes and manage the relationships between the knowledge resources.

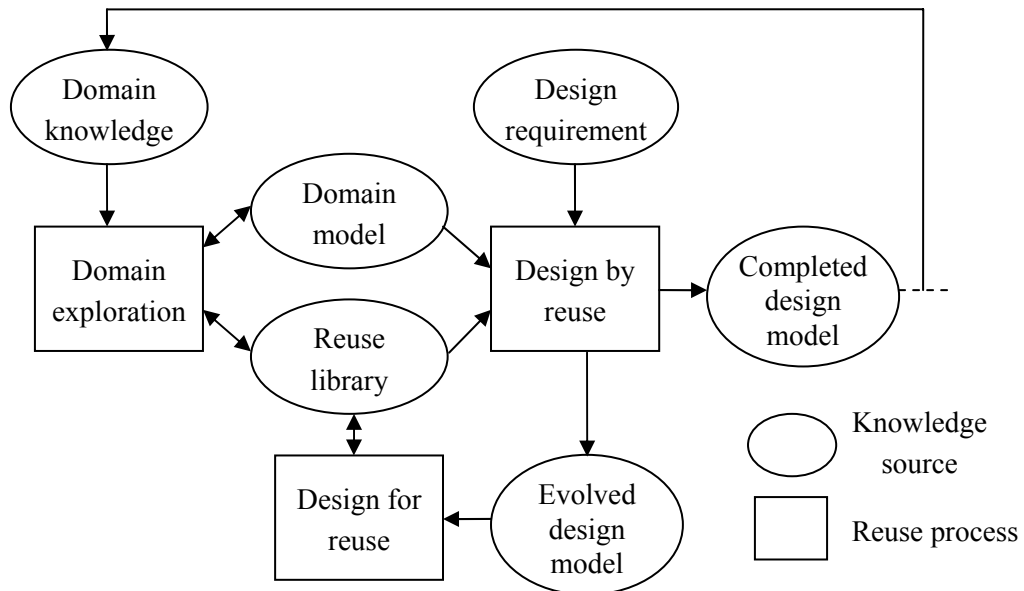


Figure 1.3 A design reuse process model (Duffy *et al.*, 1995)

### 1.2.3 Product information modeling and analysis

The representation of the product information directly influences the effectiveness of design reuse. Since the product data are inherently heterogeneous and volatile in nature, the representation scheme has to deal with information completeness, conciseness and integrity. The exchangeability of product information is also an important issue to be considered for collaborative design. Generic modeling languages, such as UML (Unified Modeling Language), CML (Compositional Modeling Language), STEP, (Standard for the Exchange of Product model data), etc., may facilitate the process. These modeling languages provide a common syntax with well-defined semantics to

model a broad variety of physical processes and objects. However, their applications have been restricted by the efficacy to deal with representation flexibility and rigor.

One important aspect of information is product function. The use of function effectively separates the design intent with the physical implementation, and hence, design is partially exempted from early engagement to specific physical structures. Function-based product design has been recognized as an effective means to conceptual design. Therefore, the representation and subsequent reasoning about function has been under extensive study (Umeda *et al.*, 1990; Iwasaki and Chandrasekaran, 1992; Gorti and Sriram, 1996; Qian and Gero, 1996; Pahl and Beitz 1996; Roy *et al.*, 2001). Relevant research issues include the representation scheme based on functions and flows, the building of function structures, the usage of taxonomy, the classification of functions, the relationships between function, form and behavior, etc.

The product information that is collected based on the above schemes is not necessarily reusable. Information is reusable if it can be easily retrieved and assembled to support solution generation. Techniques are required to transform product data into reusable forms. Thus, information analysis presents another important issue in design reuse. Information analysis usually involves the assignment of rules and the recognition of design patterns from the original data. Typical techniques include machine learning, data mining, neural networks, and heuristic methods.

#### **1.2.4 Design synthesis and evaluation**

Design synthesis refers to the generation of solutions based on reusable components. Typically, design synthesis is carried out manually, or through the interactions between humans and computers. However, to achieve efficient product design, automated design synthesis is required. Automated design synthesis is especially useful for solving large combinatorial problems, such as configuration design. Design synthesis can be carried out using various computational tools, such as agent-based methods, genetic algorithms (GA), simulated annealing (SA), branch-and-bound method, etc.

The feasibility and optimality of a design concept is assessed using the concept evaluation schemes. The major difficulty in this process is that a mathematical model is often out of the question due to the complexity of the problem. Hence, early stage solution evaluation is difficult and has been relying on intuition and experience (Ullman, 1997). Two obstacles are prominent. Firstly, evaluation usually involves multiple criteria that are inherently incommensurable. The designer can aggregate the criteria into a multivariate utility function, or alternatively, he/she can carry out the evaluation based on multiple objective optimizations. However, the multivariate utility function is not easy to formulate; and the trade-offs are hardly manageable when many objective functions are involved. Secondly, the logical management of the evaluation process is not trivial. The designer has to identify sufficient information and develop logical steps to compute the objective functions.

### 1.3 Research Objectives

The major research problem to be addressed in this research is product family design. Basically, product family design must deal with the problem of information deficiency and uncertainty. A promising idea is to collect product information from existing design cases and reuse it in new designs. However, the effectiveness of current design reuse practices is limited in the following aspects.

- (1) A comprehensive design reuse process model is lacking. Existing methods usually address one or a few aspects of design reuse. A unified approach for product family design based on the design reuse rationale is required.
- (2) Although various techniques in artificial intelligence (AI) have been proposed to extract knowledge from original data, their application in product family design is marginal.
- (3) Design reuse technologies are inadequate for solution evaluation. Comprehensive estimations based on multiple criteria such as cost and product performance are inadequate.

The purpose of this research is to develop design reuse methods to facilitate product family design. Considering the capabilities and limitations of design reuse, the research focuses on the following research issues.

Firstly, the development of a comprehensive design reuse process model that encompass the important stages of product family design. The purpose of this model is



to provide a platform to support product family design by integrating various technologies, such as product information modeling, information analysis, and intelligent solution synthesis and evaluation.

Secondly, the development of knowledge extraction techniques to identify useful information from existing products. In particular, these techniques must address issues such as: (1) the building a function-based product architecture, (2) the identification of product key characteristics (KCs) and the modularized product components, and (3) the establishment of the capabilities of the reusable components.

Thirdly, the development of product performance evaluation techniques. This involves the design of a set of uniform metrics that can incorporate diverse measures of product performance criteria into a dimensionless metric and systematic procedures to calculate the product performance by utilizing prior design knowledge.

The design reuse methodology proposed in this thesis provides ways to address the deficiencies in product family design. In particular, the problems caused by information deficiency and uncertainty can be alleviated to a certain extent through the reuse of existing product cases. The design reuse process model should enable designers to understand product family design from a holistic viewpoint. Moreover, the knowledge extraction techniques help to (1) identify useful design patterns from raw data, and (2) reformulate the information to support design reuse. Finally, the research

presents a new method, namely, the information content assessment (ICA) method, for performance evaluation. Product performance can be consistently evaluated using this method, which, in turn, enables more efficient design synthesis. Using the design reuse methodology, it is expected that improvements can be made with respect to product cost, performance and quality.

This study focuses on variant design instead of generative design. This is because the design activities involved in this method are expected to be carried out based on existing technologies. The development of new technologies and generation of innovative solutions is not covered in this study. Another implication of design reuse is that a set of existing product cases must be available. Therefore, the methods proposed in this research may not be applicable to new companies where existing products cases are not yet available.

#### **1.4 Thesis Structure**

In Chapter 2, an extensive literature review is presented. Chapter 3 proposes the framework of product family design reuse. The major elements of this framework are discussed in the subsequent chapters. Chapter 4 deals with the product information modeling and analysis. Chapter 5 presents the ICA method for product performance evaluation. Chapter 6 proposes a multiple objective optimization method to carry out the design synthesis. A prototype system to implement the design reuse methodology is

presented in Chapter 7. Three case studies are presented to show the effectiveness of the methodology. Finally, conclusions and future work are discussed in Chapter 8.

## **Chapter 2      LITERATURE REVIEW**

Two basic types of product family design approaches are discussed. The major issues of product family design are presented. A number of product family design methods and systems are discussed according to how they have addressed these issues. Based on these discussions, the limitations of the existing approaches, which signify the possible directions for further research, are pointed out.

### **2.1 Fundamentals Of Product Family Design**

The approaches to product family design can be divided into two basic types, namely, top-down and bottom-up approaches (Simpson *et al.*, 2001). The top-down approaches involve up-front decisions to develop product families based on common architectures, while the bottom-up approaches focus on the redesign and consolidation of existing products to create product families (Hernandez *et al.*, 2002). The characteristics of both types of approaches are discussed next.

#### **2.1.1 Top-down approaches**

The top-down approaches emphasize the strategic planning and design of the product platform and product family. Figure 2.1 shows the top-down approach in product family design. A product platform is developed based on the market analysis and technology advancement. Next, product variants are generated by varying the design

parameters to achieve the desired functionality. Decisions have to be made concerning the division of the market segmentations, the determination of the design specifications, the choice of the variables to control product performances, and the optimization of the design variables to achieve optimal trade-offs between commonality and performance. A product family design system has to deal with most, if not all, of these issues.

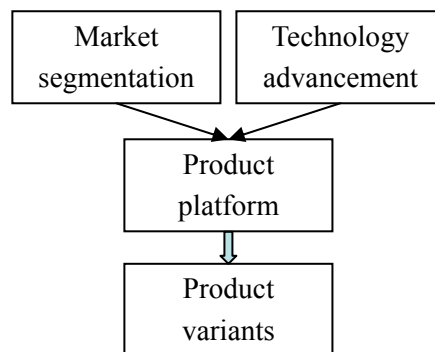


Figure 2.1 A process of top-down product family design

Among these efforts, the market segmentation grid was articulated, and the product leveraging strategies were proposed to utilize the sharing logic and cohesive architecture (Meyer and Lehnerd, 1997). A robust concept exploration method (RCEM) was proposed to build a robust product platform that can accommodate a wide range of customer requirements (Chen *et al.*, 1996). However, this is only the first step of product family design. A second step, in which products are instantiated based on the platform, is equally important. A product platform concept exploration method (PPCEM) was proposed to support scale-based product family design (Simpson *et al.*, 2001). This method explicitly defines two stages, namely, product platform design and scale-based product family design based on the platform. However, PPCEM has two

limitations. First, the commonality of the product family is determined by the designer based on a trial-and-error process. Second, the commonality is defined at only one level. In order to deal with the first limitation, a variant-based platform design methodology (VBPDM) was proposed to determine the design variables that should be made common among products (Nayak *et al.*, 2002). For the second limitation, a hierarchical platform design method was proposed to accommodate multiple levels of commonality in the product family (Hernandez *et al.*, 2002).

These top-down approaches are effective only when a product architecture can be properly defined. However, the information required to build the product architecture is immense because the dimensionality of the design space is usually high. The dimensionality of the design space refers to the number of design parameters, constraints, and objectives that have to be considered in a problem. A designer has to spend a lot of time and effort to study the intrinsic relationships between the product characteristics and the various design parameters. Since relevant information may not be available, decisions may have to be made without proper context, possibly leading to sub-optimal solutions. For example, several top-down approaches have been applied to design the universal electric motors (Meyer and Lehnerd, 1997; Simpson *et al.*, 2001; Nayak *et al.*, 2002; Hernandez *et al.*, 2002). Different strategies have been adopted to choose the design variables, and to set the feasible ranges of the variables. Accordingly, different configurations of product family have been produced, which

may not necessarily be compatible with each other. It is difficult to decide which configuration would lead to the best design practice.

### **2.1.2 Bottom-up approaches**

The bottom-up approaches depend on the analysis and reuse of products and product components. This approach is illustrated in Figure 2.2. The product platform can be established through an analysis of the existing products. Based on this product platform, new products can be developed using various design synthesis tools. Among these approaches, catalog-based design focuses on the establishment of a component catalog that can be reused in future designs based on well-indexed catalog components (Chakrabarti and Bligh, 1996; Chidambaram and Agogino, 1999). The components are usually derived from existing product cases, and are reused directly in new designs. Only simple criteria are applied for component retrieval and reusability assessment. As compared to catalog-based design, modular design is a more comprehensive method. In modular design, a set of building blocks, known as modules, is identified or created. A product family is derived by adding, removing, or substituting one or a few modules to a base platform (Pahl and Beitz, 1996). Modular design usually involves the following processes: (1) the identification of product architecture and reusable components (modules) from existing products, (2) the combination and adaptation of modules to generate new designs, and (3) the assessment of product cost and performance. A modular design system should cover all these processes. However, few systems have met this requirement.

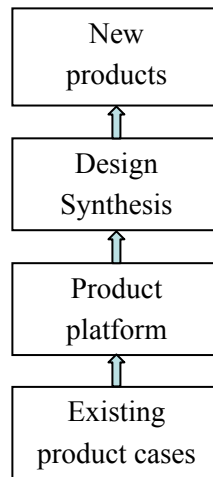


Figure 2.2 A process of bottom-up product family design

The bottom-up approaches are applied based on a set of existing products. Since the modules and product architecture are partially known, more information is available as compared to the top-down approaches. As such, information deficiency can be alleviated provided that the information of the existing products can be effectively identified. However, the bottom-up approaches have been criticized for their reliance on a large number of existing products (Hernandez *et al.*, 2002). Design freedom may be reduced if existing technologies are improperly utilized. Therefore, it is worthwhile to assess the reusability of existing products such that the design components can be logically reused and product quality ensured.

## 2.2 Design Reuse For Product Family Design

Four major issues of product family design are discussed with an emphasis on the design reuse rationale (Sections 2.2.1 ~ 2.2.4). Relevant methods to address these issues are discussed accordingly. A summary is presented in Section 2.2.5.



### 2.2.1 Representation of product information

Product information takes various forms and is subject to changes. Therefore, a comprehensive information model is required to capture the multiple facets of product information. Moreover, a proper representation scheme is necessary for the exchange and reuse of information, which is becoming more evident with the increasing collaboration across distributed design teams. A formal representation of product information has been advocated (Szykman *et al.*, 2001b). This section focuses on the formal representation of product information with an emphasis on two important issues, namely, the content of the information model and the modeling language.

Among the spectrum of product information, function has been recognized as a critical element. Hence, a function-based information model has been widely adopted in literature. Typically, function is considered as the purpose or intended use of a feature, component, or product (Ulrich and Seering, 1987; Baxter *et al.*, 1994). More specifically, function is considered as the general relationship between the input and the output of a system with the objective of fulfilling a task (Pahl and Beitz, 1996). This input-output view has been adopted and extended by many researchers (Gero, 1990; Gorti and Sriram, 1996; Kirschman and Fadel, 1998; Szykman *et al.*, 1999; Otto and Wood, 2001). A consensus is that a function is an abstraction from the physical artifacts, and hence, is not dependent on specific implementations.

Stone and Wood (2000) proposed a function basis for product design. The function basis has been motivated by several factors, such as the product architecture design, the storage and transmission of information, creativity in concept generation, etc. It enhances representation rigor and reasoning logic. Similarly, the NIST design repository project presents a generic product information model that is characterized by a formalized function-flow representation schema (Szykman *et al.*, 1998). Effort was also made to reconcile the above two approaches (Hirtz *et al.*, 2002).

However, function alone cannot accommodate the multiple facets of product information. Typical in the AI field is the inclusion of behavior in combination with function to allow for better decision-making (Umeda *et al.*, 1990; Chandrasekaran *et al.*, 1993; Iwasaki *et al.*, 1995). Behavior refers to the underlying principles or processes that make the related function attainable. At the same time, the structure/form of an artifact (features, components) is useful for an intuitive understanding of a product. Thus, function-form-behavior models have been extensively studied (Iwasaki and Chandrasekaran, 1992; Gorti and Sriram, 1996; Qian and Gero, 1996; Szykman *et al.*, 2001a; Roy *et al.*, 2001).

The above representation schemes have focused more on individual products. Recently, there is a trend towards the representation of a product family. Generic Bill-of-Materials (GBOM) was used to explore the generic product architecture and identify the assembly structure of a product family (Erens *et al.*, 1994). As an

extension of GBOM, Generic Product Modeling (GPM) was used to represent product families from the business and assembly viewpoints (McKay *et al.*, 1996). However, the functional information has been overlooked in GBOM and GPM. A Product Family Classification Tree (PFCT) was developed to model product configuration knowledge from the functional viewpoint (Yu and MacCallum, 1995). However, the interrelations between the modules and the end-products are not explicitly included in PFCT. A generic Product Family Architecture (PFA) was proposed, which explicitly deals with functional, behavioral and structural information (Jiao and Tseng, 1999). The Programmed Attribute Graph Grammar (PAGG) was developed based on the graph grammar to specify the design space and assist product family generation (Du *et al.*, 2002a). Furthermore, a graph rewriting program was developed which enables the derivation of product variants through graph transformations (Du *et al.*, 2002b). The graph grammar-based modeling is excellent in formal, visual, and extensible product family representation. However, the generation of a product family based on a few graph transformation operators falls short of being too restrictive. Moreover, the evaluation of product performance is absent. This new method requires further development to deal with more complex product family design problems.

Modeling language is another important issue in formal representation. Modeling languages allow the product information to be represented consistently and concisely. Various generic modeling languages have been employed in engineering design, such as UML (Pulm and Lindermann, 2001; Felfernig *et al.*, 2001), CML (Bobrow *et al.*,

1996), STEP (Pratt and Anderson, 2001; Szykman *et al.*, 1998), EXPRESS (Kahn *et al.*, 2001), and XML (Extensible Markup Language) (Szykman *et al.*, 1999; Rezayat, 2000). These languages provide a common syntax with well-defined semantics to model a broad variety of physical processes and objects. They are also conducive to the exchangeability, accessibility and interoperability of the product information between different design groups. However, information processing based on these languages is not straightforward due to the high level of abstraction. In product design, it is preferable to focus directly on the products, instead of the ‘entities’ or ‘physical phenomena’ that are highly abstract (Bobrow *et al.*, 1996).

### **2.2.2 Establishment of product architecture**

The establishment of product architecture is the central issue in product family design. A product architecture refers to the scheme by which the product functions are mapped to the physical components (Ulrich, 1995). Specifically, it involves “(1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components” (Ulrich, 1995). Product architecture supports product family design by developing a common technology core that addresses multiple applications. In the top-down approaches, a product architecture is built strategically without emphasis on the reasoning about existing information. Thus, the design reuse rationale is irrelevant in these approaches. Therefore, this section discusses the methods to build product architecture based on the bottom-up approaches.

In the bottom-up approaches, the building of product architecture starts from the decomposition of individual products. Basic components are identified from the functional and/or structural viewpoints. Next, the components are analyzed such that modularity and commonality are identified, which culminate in the clustering of the components into logical modules. In practice, clustering based on experience and manual operations still abound (Yu *et al.*, 2003). While this is effective for products which structures are well understood, it becomes cumbersome for complex products. Rapid and intelligent tools are required to facilitate modularity analysis and product architecture building.

Modularity in product family design has constantly been represented as matrix operations. Huang and Kusiak (1998) proposed an approach to represent modularity and develop generic modular products. Gu and Sosale (1999) developed a modularization method to enhance modularity from different design perspectives. A similar rationale was adopted in the framework of the House of Modular Enhancement (HOME) (Sand *et al.*, 2002). However, these two methods were aimed at individual products instead of a product family. Martin and Ishii (2002) proposed the design for variety (DFV) method to build modular product architecture for multiple generations of products. The DFV method uses the Quality Function Deployment (QFD) to identify modules within a product family. The QFD has also been adopted and extended by a few other methods (Erixon, 1996; Ericsson and Erixon, 1999; Sand *et al.*, 2002). The design structure matrix (DSM) is another effective tool to determine

product modularity (Dong and Whitney, 2001; Yu *et al.*, 2003). One limitation of the above methods is that they have not employed a formal representation of the product information.

A formal representation of product functions enhances the establishment of product architecture. Quantitative methods have been proposed to represent product function, and identify product architecture based on functional interdependence and product similarity (McAdams *et al.*, 1999; Stone *et al.*, 2000a). Heuristic methods have been proposed to identify architecture and modules based on a set of heuristic rules (Zamirowski and Otto, 1999; Stone, *et al.*, 2000b). An analytical method was proposed to incorporate the customer demands into architecture building (Yu *et al.*, 1999). Moreover, a modular product architecture was developed to permit the platform to shift in size and type (Dahmus *et al.*, 2000). These are representative approaches to establish product architecture based on the formal representation of function. However, the application of the quantitative methods and heuristic rules still relies heavily on human designers, which hinders their efficiency and consistency. Hölttä *et al.* (2003) proposed an algorithm to generate a modular product architecture based on the metric of module distance. This method uses quantitative measures and is supported by computational tools. Finally, computational tools, such as SA (Gu and Sosale, 1999) and GA (Yu *et al.*, 2003), have been adopted to build modular product architecture.

Despite these efforts, the establishment of a product architecture still needs to be enhanced. First, the establishment of such an architecture should be consistent with the product information modeling schemes. Formal representation schemes should be used to build product architectures. Second, computational tools are required to enable rapid and intelligent building of product architecture.

### **2.2.3 Product family design as a configuration design problem**

From the reuse perspective, product family design is a process of synthesizing product configurations from existing components based on the product architecture. This includes direct retrieval of relevant modules or synthesis of products by combining a set of modules. The process can be carried out manually or automatically.

Configuration design features a broad variety of design problems. Mittal and Frayman (1989) defined the configuration task as the selection and combination of predefined components which satisfies a set of requirements and constraints. This definition has been widely adopted and extensively investigated in literature (Yu and MacCallum, 1995; Wielinga and Schreiber, 1997; Corbett and Rosen, 2004). In engineering design, proper formulation of the design problem and effective exploration of the design space are the two fundamental issues.

Siddique and Rosen (2001) proposed the Product Family Reasoning System (PFRS) to formulate product platform design as a configuration design problem. A set of

viewpoint-specific design spaces was combined into a 'common' product variety design space, where constraints were applied to extract the feasible design regions. As an extension to PFRS, a partitioning method was proposed to reduce the size of the feasible design space (Corbett and Rosen, 2004). However, there is an implicit assumption in both methods that the feasible space is small enough to be easily manipulated, leaving the selection of suitable/optimal solutions to the designers. In cases where the feasible design space is large, more effective search algorithms are required. Fujita *et al.* (1999) proposed a modular design approach for product family configuration design, where SA was used to search for the optimal solutions. Ong *et al.* (2006) formulated product customization as a Constraint Satisfaction Problem (CSP), which was solved using the invasion-based algorithm. Gonzalez-Zugasti and Otto (2000) used GA to search for the optimal solutions in a module-based approach. Sabin and Weigel (1998) investigated rule-based, model-based and case-based approaches to configuration design.

Recently, agent-based methods have received more attention in configuration design. Campbell *et al.* (1999) proposed an A-Design approach to explore the design space in search of the Pareto-optimal solutions. However, A-Design was not aimed at product family design. Rai and Allada (2003) developed a module-based product family design system that combined multi agent systems, function architecturing and multi-objective optimization. Liang and Huang (2002) investigated the role of intelligent agents in satisfying customer requirements using a collaboration information system.



Agent-based design offers a promising solution to rapid and concurrent product configuration design (Shen *et al.*, 2001).

#### **2.2.4 Optimization and solution evaluation**

As mentioned earlier, the major concern of product family design is the management of trade-offs between commonality and performance. Therefore, the effectiveness and efficiency of product family design should be evaluated according to these two criteria. Commonality is a vague term that is closely related to production cost, under the assumption that maximizing commonality among products minimizes cost (Simpson, 2004). However, for more accurate estimations, it is preferable to model the market demands and the associated manufacturing cost directly.

##### **2.2.4.1 Cost estimation**

Design and production cost can be estimated using qualitative or quantitative measurements. Qualitative measurements provide rough guidelines to manage the product family. For example, Martin and Ishii (1996, 1997) used several indices, namely, commonality index, differentiation index and setup index, to measure the cost of product variety. On the other hand, quantitative measurements allow for more accurate estimation and better control of solution optimality. The cost models thus developed have been dependent on the estimation of various cost elements, such as materials, machine time, direct labor, overhead, etc. Many cost models have been proposed, such as function costing (Hundal, 1997), magnitude-based costing (Hundal,

1997), and activity-based costing (Cooper and Kaplan, 1991). A common deficiency of these cost models is their reliance on a detailed knowledge of product design and process plan, intertwined with the uncertainty of such knowledge at the early design stage (Jiao and Tseng, 2004). These have made cost estimation a formidable task. To overcome this difficulty, cost estimation based on prior knowledge and careful decomposition of cost elements is necessary.

#### **2.2.4.2 Product performance evaluation**

Evaluation of product performance is necessary to ensure that the product family can satisfy the customer requirements. Two issues are critical: (1) how to establish the relationship between the performance and the design parameters, and (2) how to accommodate the diverse performance criteria that are inherently incomparable.

For the first issue, the relationship can be considered as a mapping route from performance to design parameters. A straightforward method is to establish equations between the input variables and the output performance (Nelson *et al.*, 2001; Chidambaram and Agogino, 1999; Simpson *et al.*, 2001). When such equations can be established, the estimation can be accurate. However, in a typical engineering problem, it is difficult, if not impossible, to formulate appropriate equations (Simpson *et al.*, 2001). The presence of uncertainties, variations, and noise factors worsens the problem (Ulrich and Eppinger, 2004). Alternatively, meta-models can be used to simulate the equations. The meta-modeling techniques that have been used in product family design

include, design of experiments (DOE) (Simpson, 1998; Ulrich and Eppinger, 2004), response surface method (RSM) (Chen *et al.*, 1996; Simpson *et al.*, 1996; Jiao and Tseng, 2004), regression analysis (Fujita and Yoshioka, 2003), etc. The prerequisite of meta-modeling is that experimental data can be obtained from existing cases or from simulation results. However, it is not always possible to satisfy this condition. If invalid experimental data are used to build the models, these models can be invalid. The correlations between the input parameters and the output performance can also be approximated using the QFD method (Martin and Ishii, 2002). Accordingly, the output performance can be estimated based on the principles of design-by-analogy.

For the second issue, it is desirable to aggregate diverse measures of design performance into a unified metric. A possible solution is to normalize the multiple performance criteria and aggregate them into a utility function (Simpson *et al.*, 2001; Hernandez *et al.*, 2002; Mangun and Thurston, 2002). However, the inherent incompatibility between different criteria and the arbitrary assignment of weights to these criteria hinder the effectiveness of such utility functions.

Recently, there is an advocate to use information content as a unified measurement of multiple criteria. Information content is defined in terms of the probability that design requirements can be satisfied by the design parameters (Suh, 2001). It is a dimensionless metric that can be computed using statistical techniques. The major concern is how to formulate the problem and compute the information content.

Information content was calculated with the support of fuzzy QFD, and was used to estimate the critical characteristics of routine design (Bahrami, 1994). The limitation with this method is that the QFD process could only roughly estimate the customers' preferences. A concept evaluation method based on information content and fuzzy ranking was used in configuration design for mass customization (Jiao and Tseng, 1998). An appealing feature of this method is that it combines 'tangible' and 'intangible' criteria into the same evaluation metric. Information content was used to measure design customizability in mass customization, where the system ranges and design ranges, which are necessary to compute the information content, were assumed to be known (Jiao and Tseng, 2004). However, it is not really easy to obtain the system ranges and design ranges. Actually, information content as an evaluation metric has not been widely adopted, mainly due to the difficulties to formulate the design ranges and system ranges. Therefore, proper procedures to derive these two ranges have to be developed in order to apply the theory.

### **2.2.5 Look back and look ahead**

Based on the discussions, the product family design approaches and their support to the four major issues are summarized in Table 2.1. These approaches are roughly divided into five categories, namely, scale-based, model-based, graph-based, module-based, and others (which include a few miscellaneous approaches). Each method is examined with respect to its support to information modeling, information

analysis (mainly focusing on the building of product architectures), design synthesis and solution evaluation.

The scale-based approaches are based on the top-down design rationale, which does not emphasize the utilization of existing product information. Therefore, information modeling has been absent in these approaches. The building of product architecture and the design synthesis and evaluation have been formulated into a combined process, typically supported by computational tools. Multi-objective optimization has been used in performance evaluation, and the performance has been correlated with the design variables using equations or meta-models, such as DOE and RSM. On the other hand, configuration design is not relevant in these methods. Cost has rarely been considered as an optimization objective, maybe because of the lack of information to make up-front estimations. Actually, the lack of information is characteristic of the top-down approaches, which have to strive hard to deal with the high dimensionality of the design space. This is where design reuse can play an important role.

Table 2.1 Summary of product family design approaches

Approach (* indicates that the approach is aimed at the design of individual products instead of a product family)		Information modeling					Information analysis					Design synthesis			Solution evaluation					Application		
		Content				Taxonomy	Modeling language	Style				Tool   algorithm	Style			Objectives	Cost estimation	Performance				
		Function	Behavior	Structure (form)	Interface			None (assumed)	Manual	Heuristic rule	Computational		Configuration design	Manual   case retrieval	Automated synthesis			Algorithm	Qualitative preference		Equation	Meta-modeling
Scale-based	D'Souza & Simpson (2003)								x	GA			x	GA	M				DOE	x	Aviation aircrafts	
	Fujita & Yoshioka (2003)								x	GA			x	GA	M	x			RSM	x	Lift-gate dumpers	
	Hernandez <i>et al.</i> (2002)								x	LPP			x	LPP	M				RSM	x	Universal electric motor	
	Nayak <i>et al.</i> (2002)								x	SLP			x	SLP	M			x		x	Universal electric motor	
	Simpson <i>et al.</i> (2001)								x	GRG			x	GRG	M			x		x	Universal electric motor	
Model-based	*Campbell <i>et al.</i> (1999)	x	x	x	x	x					x		x	SAGA	M	x		x			Weighing machine	
	Chidambaram & Agogino (1999)	x					x				x		x	GA	S	x					Brushless DC motors	
	*Counsell <i>et al.</i> (1999)	x	x	x		x	O-O				x	x									Factory automation	
	*Duffy <i>et al.</i> (1996)		x	x			O-O		x	ML											Pump	
	*Szykman <i>et al.</i> (1998)	x	x	x	x	x	XML				x	x									Cordless screwdriver	
Graph-based	Corbett & Rosen (2004)	x			x		GRA		x	CCP	x		x	PT							Automotive underbodies	
	Du <i>et al.</i> (2002a)	x			x		GRA		x	PAGG	x	x		GR							Power supplies	
	Du <i>et al.</i> (2002b)	x	x	x	x		GRA	x			x	x		GR							Office chairs	
	Siddique & Rosen (2001)	x			x		GRA		x	CCP	x		x	CCP							Coffeemakers	
	Dahmus <i>et al.</i> (2000)	x				x		x	x	MM											Cordless screwdrivers	
	Du <i>et al.</i> (2001)							x			x	x					x				Power supplies	



The model-based approaches emphasize the modeling of existing products. The multiple facets of product information have been carefully dealt with and supported by generic modeling languages. Design catalogs and design repositories have been built to make the information easily accessible by different parties. However, these methods have focused more on the design of individual products than on product families. Therefore, the support to build product architecture is inadequate. Design synthesis has been supported by various techniques. For example, A-Design used the agent-based tool that combined SA and GA for intelligent configuration generation (Campbell *et al.*, 1999). In the NIST design repository project (Szykman *et al.*, 1998), design synthesis was restricted to manual retrieval of product cases.

The graph-based approaches are distinctive for providing support to product modeling and reasoning. In particular, the graph-based approaches have used graph representations to capture the multiple facets of product information in mathematical forms. Moreover, product architecture building and subsequent design synthesis have been supported by predefined graph grammar. However, the graph transformation operators are restrictive for solution generation. Another restriction is the absence of performance estimation, which can be attributed to the lack of operators to establish the relationships between design variables and performance.

A number of product family design methods are included in the module-based category. Module-based methods provide extensive solutions to the major issues of product



family design. Among them, the function-based product information modeling has been widely adopted. Moreover, the building of product architecture has been supported by various analytical techniques such as heuristic rules or computational algorithms. However, the process usually involves extensive human operations and relies heavily on expertise, making it inefficient and not repeatable. Various algorithms have been applied to configuration generation, such as exhaustive search, nonlinear programming, and derivative-free methods (e.g., GA and SA). GA has been advocated by many researchers due to its power in solving large combinatorial problems, typical in product configuration design.

As can be seen from the above analysis, currently there are no approaches which can address all the issues in product family design reuse. In the development of a comprehensive design reuse process model, existing methods usually address one or a few aspects of design reuse. A unified approach for product family design based on the design reuse rationale is absent. In order to achieve effective design reuse, it is necessary that the design process be organized logically to support domain exploration, design-for-reuse, and design-by-reuse. In the development of information processing techniques, existing methods lack scientific rigor in building the function-based product architecture, which is indispensable to define the mapping schemes from the customer domain to the design domain. Hence, it is important to develop intelligent tools to extract product architectures from raw product data. In the development of product performance evaluation techniques, there is a lack of uniform measurement

and consistent evaluation procedures. To allow for rapid product family configuration design based on computational tools, a uniform measurement and consistent evaluation procedure has to be defined.

Considering these deficiencies, this thesis attempts to address the following research issues.

- (1) Development of a comprehensive design process model that can integrate information modeling, information processing, design synthesis and evaluation.
- (2) Establishment of a function-based product architecture based on a formal representation of product functions and intelligent tools that enhances the efficiency and repeatability of the process.
- (3) Adoption of information content as a uniform metric that accommodates the diverse performance criteria; development of logical procedures to compute product performance based on the information content.
- (4) Formulation of the design synthesis problem; application of efficient optimization algorithms to solve the design synthesis problem.

### **2.3 Summary**

This chapter presents an extended literature review of the methodologies and systems of product family design, with an emphasis on the role of design reuse. Product family design has been supported by (1) the function-based product information model, (2) analytical and heuristic methods to build product architecture, and (3) various solution

synthesis methods based on optimization algorithms. However, few systems have effectively combined these techniques, i.e., they have focused on one or a few aspects of product family design. It is desirable to develop a system that integrates these techniques to achieve more efficient information reuse. In Chapter 3, a comprehensive design reuse process model is proposed to achieve this.

# **Chapter 3      FRAMEWORK OF PRODUCT FAMILY DESIGN REUSE**

A comprehensive design reuse process model for product family design is proposed, and it establishes the foundation of the product family design reuse (PFDR) methodology. This process model incorporates information modeling, product platform building and product family generation and evaluation. This chapter provides an overview of the methodology. First, the design reuse process model is introduced in Section 3.1. Next, a few hypotheses are presented, and the problem boundaries are defined in Section 3.2.

## **3.1      Integrated Design Reuse Process Model**

The PFDR methodology defines systematic procedures to design product family. Design reuse is achieved by identifying reusable product architecture and components, followed by simultaneously designing a set of products using automated solution synthesis and evaluation. The process model includes the steps and tools to implement the design reuse methodology. Basically, it involves three major stages, namely, product case modeling, knowledge extraction, and design synthesis and evaluation (Figure 3.1). The functions and associated tools of each stage are discussed next.

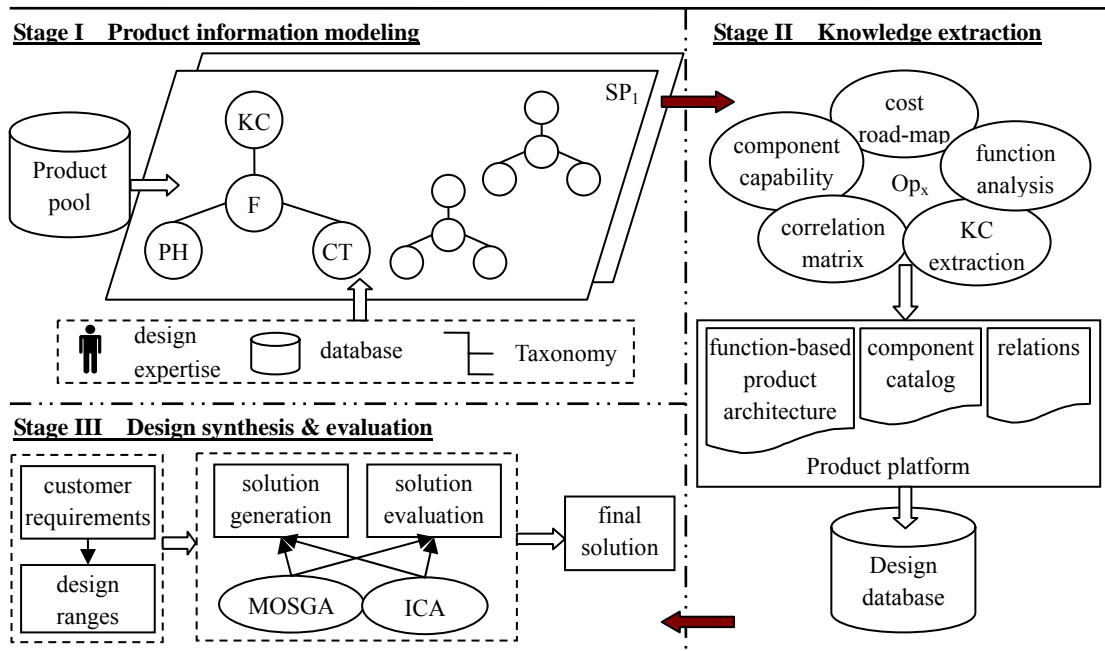


Figure 3.1 The PFDR process model

KC: key characteristics      F: functional      PH: physical      CT: contextual

SP<sub>i</sub>: a general design space      Op<sub>x</sub>: knowledge extraction operators

MOSGA: multi-objective struggle genetic algorithm      ICA: information content assessment

### 3.1.1 Stage I: Product information modeling

Design reuse begins from the modeling of the product cases that are available from previous designs. The sources of product cases include past models of the products in a company or those of the competitors. Similar products that share certain commonalities are decomposed within a general design space, denoted as SP<sub>i</sub>. Thus, a general design space contains similar products that are expected to form a product family. The decomposition is carried out with respect to a predefined information model, which accommodates the multiple facets of product data, namely, the key characteristics (KCs), and the functional, physical and contextual information. The

KCs are qualitative or quantitative customer requirements that are translated to support engineering specifications for product design (Rezayat, 2000). The content of the other facets will be discussed in Chapter 4. Thus, within a  $SP_i$ , there are a number of individual product cases that share a commonality to a certain extent. At this stage, the innate commonality, the design patterns, and their impact on designing the product family are not evident yet. The individual cases are considered as a set of loosely related information entities, and hence are called raw product data. The decomposition of a product into multiple facets is the prerequisite for establishing a modular product architecture.

Information modeling is supported by various techniques and knowledge sources, such as design expertise, database, and taxonomy. In particular, the taxonomy is used to enhance the representation rigor of product functions.

### **3.1.2 Stage II: Knowledge extraction**

At this stage, a set of analytical techniques is used to extract knowledge from the raw product data. The aim of knowledge extraction is to identify design patterns and transfer the relevant information into reusable forms, which facilitate the building of a product platform. The analytical techniques are collectively called knowledge extraction operators, and denoted as  $\{Op_x\}$ . The following operators are developed.

### **3.1.2.1 Function analysis ( $Op_f$ )**

Function analysis is responsible for establishing the function-based product architectures (FPA). The prerequisite for function analysis is that the function structures of a set of individual products have been built in Stage I of the PFDR model. The FPA is generalized from these individual function structures. It contains the common functions of a product family, and provides a platform upon which the new products can be built. This research proposes a new method, namely, the self-organizing map (SOM), to facilitate this process. The procedures to carry out  $Op_f$  and the advantages of the SOM method in comparison with the traditional methods are discussed in Chapter 4.

### **3.1.2.2 KC extraction ( $Op_k$ )**

The KCs that are common to a group of products within the same general design space are extracted. They are used to limit the scope of the problem by focusing on a set of crucial business or customer factors that defines the spectrum of the product family. KC extraction is carried out by the senior designers who are familiar with the market and customer requirements.

### **3.1.2.3 Correlation matrix ( $Op_r$ )**

A mapping route from the design parameters to the major product performance is established. The design parameters are the different attributes of the physical components. As an objective of the design reuse system, the performance of the

product family will be optimized with respect to cost and various design constraints. To do so, the performance of the product family has to be estimated based on the attributes of the physical components. Establishing the relationships between the performance of the product family and the design parameters is necessary for performance evaluation. In this research, the correlation is established as a few correlation matrices similar to the house of quality in the QFD method.

#### **3.1.2.4 Cost modeling ( $Op_c$ )**

A cost model of the product family is established and the cost road-maps of the components are identified. Various cost elements can be considered in the cost model. In design reuse, the model includes the cost elements at three levels, namely, the corporation/department level, the product level, and the component level. Analytical tools are used to determine these different cost elements. The cost road-maps are established for components based on existing product cases or quotations from the OEMs.

#### **3.1.2.5 Component capability index ( $Op_i$ )**

The component capability index refers to the extent to which a component can satisfy the related performance requirements. The capability indices are defined for individual components or component combinations. It is useful in the performance evaluation using information content. The capability indices are extracted from existing products. Chapter 5 proposes the procedures to establish the capability indices and use them to



evaluate product performance.

Using these knowledge extraction operators, a product platform is constructed and a component catalog is built. The relationships between the different aspects of the product information are established. These constitute the design knowledge base that can be accessed by different designers.

### **3.1.3 Stage III: Design synthesis and evaluation**

The last stage involves design synthesis and evaluation, in which the configurations of a set of related products are generated, and the production cost and product performance are evaluated. Basically, design synthesis and evaluation is formulated as a multi-objective optimization problem, and solved using a multi-objective struggle genetic algorithm (MOSGA) (Andersson and Wallace, 2002). First, a ranged set of design requirements are interpreted into design ranges with respect to the KCs. Next, the product configurations of the product family are automatically generated, evaluated, and updated using the GA-based search. Evaluation is performed according to two criteria, namely, cost and performance. Cost is estimated using the cost model developed in Stage II (Figure 3.1). Performance is evaluated as the average information content of a product family. The use of information content is a distinctive feature of this research. After the optimization process, the final solution can be chosen from the Pareto-optimal set based on the designer's preferences.

The PFDR methodology establishes a consistent framework to facilitate product family design based on the reuse rationale. It emphasizes the application of AI techniques in the establishment of product platforms and the generation of product family. However, to carry out these procedures, the design problem has to be formulated with some restrictions. This is universal to engineering design in the sense that the problem formulation inevitably reduces the design space (Fujita and Yoshida, 2001). Therefore, a few hypotheses and the definition of the problem boundary are presented next.

## **3.2 Prerequisites And Problem Boundaries**

### **3.2.1 Prerequisites**

#### **Prerequisite 1 – Product case availability**

This prerequisite is related to the applicability of design reuse, which has been discussed in the introductory part of Chapter 1 (Page 3). It is assumed that a group of designed product cases is available and their properties are known. The existing product cases are the source of information to be reused. Thus, it is desirable that the proposed method be applied in an established company where detailed information of existing products is accessible. Otherwise, the collection of products through market investigation becomes necessary. Engineering techniques, such as reverse engineering, may be required to collect the relevant information. In the case studies to be presented in Chapter 7, this prerequisite has been satisfied through different ways. Case study I (mobile phone product family design) and case study III (fan filter unit design) have been carried out in established companies, where a number prior design cases are

available. In case study II, the initial product information is extracted from the reported work. In all cases, this prerequisite can be satisfied.

### **Prerequisite 2 – Product architecture modularity**

The method is effective when the product has a modular structure. This requirement arises from the need to formulate the design synthesis as a configuration design problem. For products that have an integral architecture, many product functions are coupled, and hence, it may be difficult to identify reusable components and deal with the incompatibilities among the reused components. Therefore, the method is not applicable to products with an integral architecture. Fortunately, a modular structure has been adopted in many products, such as home electronics, power tools, automobiles, etc. Thus, the design reuse method is not restricted to specific products. The case studies presented in Chapter 7 deal with three types of products, namely, mobile phone, TV receiver circuits, and fan filter units. All these products assume a modular structure. Hence, the design reuse methodology can be applied in these case studies.

### **3.2.2 Problem boundaries**

#### **Variant design versus generative design**

This research focuses on variant instead of generative design. This is an implication of the first prerequisite. In particular, the design activities involved in this method are

expected to be carried out based on existing technologies. The generation of innovative solutions and exploration of new technologies were not covered in this study.

### **Module combination optimization versus module attributes optimization**

In this research, design synthesis is formulated as a configuration design problem based on predefined modular product architectures. This involves an optimization process that focuses on the optimality of module combinations. In product family design, the optimization of module attributes is another important issue (Fujita and Yoshida, 2001). However, the module attributes are largely fixed according to the rationale of design reuse. Therefore, the optimization of module attributes is not dealt with in this research.

# Chapter 4 ESTABLISHMENT OF PRODUCT PLATFORM

This chapter presents the methods involved in Stages I and II of the PFDR methodology. A function-based product information model is used to decompose information of the product cases into multiple facets. Based on the formal representation schemes, automated clustering of product functions is carried out using neural networks techniques, leading to the establishment of the FPA. The other knowledge extraction operators, namely, the KC extraction ( $Op_k$ ) and correlation matrix ( $Op_r$ ) are presented. These efforts contribute to the establishment of a product platform for product family design.

## 4.1 Function-Based Product Information Model

The product information model adopted in this research involves multiple facets of product data. The data structure of the information model is presented in Section 4.1.1. Product function is modeled using the flow-based, vector representation schemes (Section 4.1.2). The function model is supported by taxonomy (Section 4.1.3).

### 4.1.1 Product information representation

A product case  $p_i$  is represented as a 4-tuple, denoted as:

$$p_i \square (F_i^0, K_i^0, M_i^0, X_i^0)$$

where:

- (1)  $F_i^0$  denotes a hierarchical function structure, which is obtained through function decomposition. This research adopted the Function Analysis System Technique (FAST) method to carry out the function decomposition process (Otto and Wood, 2001). The decomposition stops when each of the functions can be fulfilled by a single, basic solution principle. Such a function is called an **atomic function**. Thus, a flow-oriented, hierarchical function structure is established for each product case. Two types of relationships may exist between the functions, namely, descendant and communication. The former refers to whether a function has one or a few sub-functions (children). The latter describes two functions that are connected by flow(s): a function can either be the source or the destination of another function according to the direction of the flow(s). Figure 4.1 shows the data structure of function and flow.

```

TFunction // TYPE: function
{
    name: String;      //name of the function
    input: TFlow;     //input flow
    output: TFlow;    //output flow
    action: String;   //a verbalization of function
    sub: TFunction;   //subordinate/child function block
}

TFlow // TYPE: flow
{
    type: Enumeration; //type of flow (ENG, MAT, SIN)
    name: String;      //unique name of a flow
    source: TFunction; //function block as the source of flow
    dest: TFunction;  //function block as the destination of flow
}

```

Figure 4.1 Data structure of function and flow

(2)  $K_i^0 = [k_{i1}, k_{i2}, \dots, k_{ip}]^T$  is a vector of the KCs that signify the performance features of  $p_i$ . They are qualitative or quantitative engineering specifications that are related to the customer requirements. The data type of a KC can be one of the following: categorical, ordinal, Boolean, and real number. Among them, the real number is continuous while the other three are discrete. The categorical data represents a set of mutually exclusive values. For example, the material of a part can be ‘stainless steel’, ‘aluminum’, ‘plastic’, etc. A Boolean type refers to whether a characteristic is present or not, and the value of a Boolean type KC is restricted exclusively to ‘True’ and ‘False’. An ordinal type is similar to the categorical type except that it has an ordering feature, i.e., a higher order value can satisfy a lower order requirement. For example, the service cleanliness of a clean room can have the following grades {100000, 10000, 1000, 100, 10, 1}. The succeeding ones can satisfy the preceding ones but not *vice versa*.

```

TKC // TYPE: Key characteristics
{
  name: String;
  type: Enumeration; //4 types: categorical, Boolean, ordinal, real
  value: Variant;
}

```

Figure 4.2 Data structure of KCs

(3)  $M_i^0 = [m_{i1}, m_{i2}, \dots, m_{in}]^T$  contains the physical components that implement the corresponding product functions. Each physical component ( $m_{ij}$ ) implements one or a few atomic functions. The physical component is basically a geometric model, preferably a CAD model. The attributes of the physical component are included,

such as material, dimension, cost, weight, etc.

```

TPhysical // TYPE: physical (geometric model)
{
  name: String;
  ref_fun: TFunction; // related function.
  att: TAttribute; //attributes of the physical component.
  ref_model: String; //reference to the physical model.
  modeler: Enumeration; // CAD modeler.
}

```

Figure 4.3 Data structure of physical components

(4)  $X_i^0$  represents the contextual information. It captures the relationships between the different information facets, namely, (1) the relationships between KCs and functions (denoted as  $X_{K-F}$ ), which describe whether a KC is dependent on one or a few functions, and (2) the interdependency relationships between different physical components (denoted as  $X_{M-M}$ ), which refer to whether two components are compatible with each other.

```

TContextual // TYPE: Contextual
{
  name: String;
  kc_f: TMap; // mapping relation between KC and function.
  m_m: TCom; // compatibility relation between physical components.
}

```

Figure 4.4 Data structure of contextual information

Each product case is analyzed and modeled according to the data structures. Similar products are assigned to the same general design space ( $SP_i$ ) to be further analyzed using the knowledge extraction operators. Among these operators, function analysis



( $Op_f$ ) is the most important operator.  $Op_f$  requires a formal representation of functions, which is discussed next.

#### 4.1.2 The key element vector representation of function structure

Based on the flow-oriented, hierarchical function structure, an atomic function is defined in terms of the input/output flows and the function actions, i.e., the input flows are transformed into the output flows as a result of the action of the function (Pahl and Beitz, 1996). A flow can be one of the three forms, namely, energy ( $E_{NG}$ ), material ( $M_{AT}$ ) and signal ( $S_{IN}$ ), which are defined in the flow taxonomy. An action of a function is in the form of a transitive verb that is defined in the function taxonomy. Based on these definitions, each atomic function is described as follows, where [] denotes optional object(s):

transitive verb(s) + [the input flow] + [the output flow]

For example, an electric kettle must have a function of heat generation. A ‘heat generation’ function may involve ‘converting’ the input energy ‘electric wattage’ to the output energy ‘heat’. As a result, the function can be represented as: *convert electric wattage to heat*. Figure 4.5 shows a block representation of the function.

Thus, an atomic function is formally represented as three main attributes, namely, the input flow ( $I_W$ ), output flow ( $O_W$ ) and the function action ( $A_F$ ). Each function action, in

combination with the corresponding input/output flows, constitutes a key element  $\kappa$ .

Thus, a key element is represented as a vector:

$$\kappa = [A_F \quad I_W \quad O_W] \quad (4.1)$$

where:

$I_W$  is the input flow(s), and  $I_W \in E_{NG} \cup M_{AT} \cup S_{IN}$ , and

$O_W$  is the output flow(s), and  $O_W \in E_{NG} \cup M_{AT} \cup S_{IN}$ .

If a function involves multiple function actions and multiple key elements, this function is represented as a key element vector (KEV), denoted as  $\vec{f}$ :

$$\vec{f} = [\kappa_1 \quad \dots \quad \kappa_m]^T = \begin{bmatrix} A_{F1} & I_{W1} & O_{W1} \\ \dots & \dots & \dots \\ A_{Fm} & I_{Wm} & O_{Wm} \end{bmatrix} \quad (4.2)$$

where  $m$  is the total number of function actions. Usually  $m$  should not be a large value because in such a case, the function can be further decomposed into sub-functions. As an example, the ‘heat generation’ function can be represented as a one-element KEV:

$$\vec{f} = [\text{convert} \quad \text{wattage:electric signal} \quad \text{heat}].$$

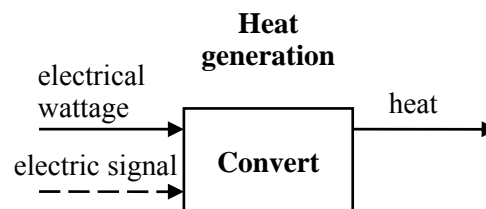


Figure 4.5 A block representation of function - ‘heat generation’

Finally, assuming that the characteristics of non-atomic functions can be defined by the complete set of its descendant atomic functions, the function structure  $F_i^0$  of product  $p_i$  can be converted into the KEV format:

$$F_i = [\bar{f}_1 \quad \bar{f}_2 \quad \dots \quad \bar{f}_M]^T \quad (4.3)$$

where  $M$  is the total number of atomic functions.

The elements in a KEV are non-numeric, and as such they are not suitable for computational analysis. The solution to this problem involves two steps: firstly, developing a taxonomy with proper coding schemes; and secondly, mapping the function actions and flows to quantitative values using these coding schemes.

### 4.1.3 Function and flow taxonomies

The purpose of using taxonomy is to represent the typical product functions with a limited number of vocabularies, which are “as small as possible, yet generic enough to allow modeling of a broad variety of engineering artifacts” (Szykman *et al.*, 1999). Pahl and Beitz (1996) divided the input/output flows into energy, material and signal, and developed a set of generally valid functions in the form of verbs (e.g., change, vary, connect, etc.). Otto and Wood (2001) refined Pahl and Beitz’s taxonomy with eight categories of functions (channel, support, connect, branch, provision, control magnitude, covert, signal) and three categories of flows (energy, material and signal). However, the taxonomy includes various ‘synonyms’ and ‘compliments’, which makes it complicated to be transformed into numerical forms. Kirschman and Fadel (1998)

classified the vocabularies into four groups, namely, motion, power/matter, control and enclosure. They were further arranged in a sentence form leading to approximately 150 combinations of elementary mechanical functions. However, the sentence form is not suitable for developing a rigorous function model. The taxonomy adopted by Szykman *et al.* (1999) consists of six function categories and three flow categories. The categories adopt a multi-level structure that has dual effects. On the one hand, it makes the taxonomy generic and flexible, and on the other hand, it presents difficulties for computational processing. Another possible solution is to define consistent and unique function taxonomies without distinguishing function and flows. However, such a practice is limited in two perspectives. Firstly, it is very difficult to develop such a set of function taxonomies for different types of products. In other words, while it is possible to develop function taxonomies for a set of similar products, developing function taxonomies for a variety of products is not easy. It may result in a huge set of vocabularies that compromise the initial intent of developing taxonomies, namely, to describe product functions using a small set of vocabularies to allow modeling of a broad variety of engineering artifacts (Szykman *et al.*, 1999). Secondly, even if such a set of function taxonomies can be developed, building the function structures for the products becomes a challenging task because the user has to select the vocabularies from a huge set of taxonomies. This is not different from building product architectures based on expertise or empirical observations, which is not efficient and lacks scientific rigor.

The taxonomy in this research is extracted and refined from the existing research. In particular, the flow taxonomy is defined for three main types, namely, energy, material, and signal, with one level of sub-category in each type. The function action taxonomy includes four basic types. Three types are defined with respect to the three flow types; and the last type deals with the assembly and spatial relations of product components, known as the ‘enclosure’ functions. Each of the first three categories involves one level of sub-category. In comparison to the reported work, the taxonomy adopted in this research has a simpler structure, such that they can be easily coded. Figure 4.6 shows an excerpt of the function and flow taxonomy. Detailed taxonomy tables are included in Appendices A and B.

The taxonomy is further coded such that a function action is represented by a unique 3-digit code and a flow is represented by a unique 4-digit code (Figure 4.7).

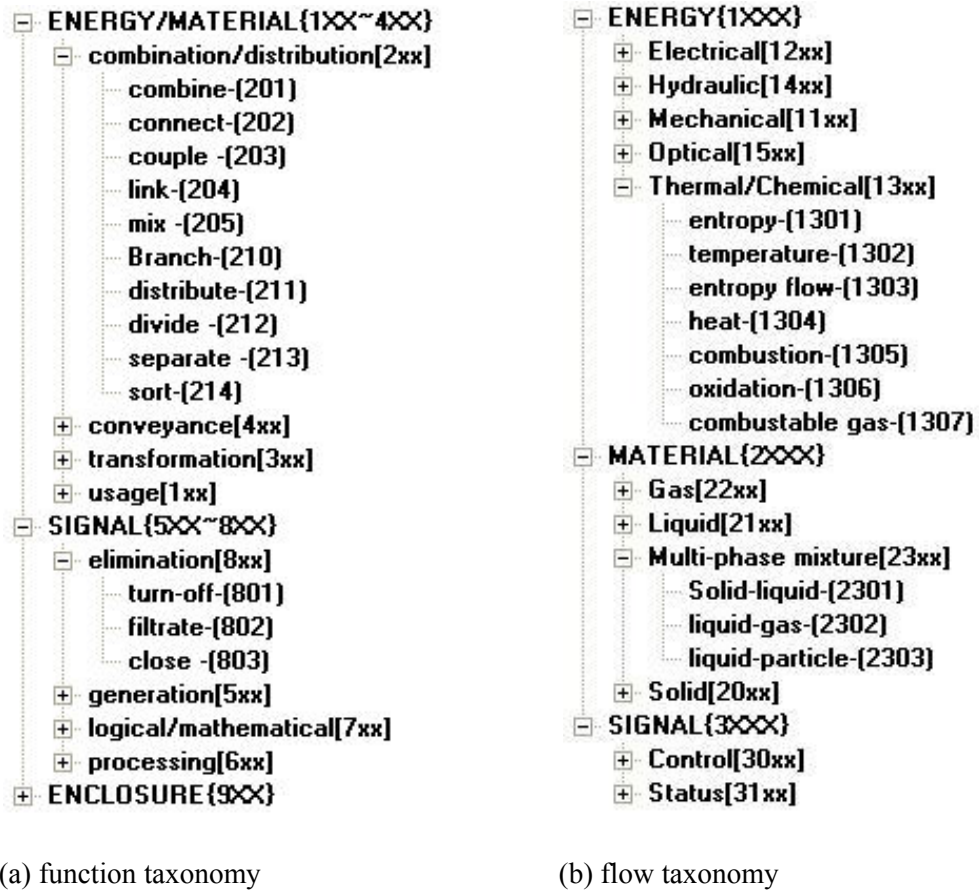


Figure 4.6 An excerpt of function action and flow taxonomies

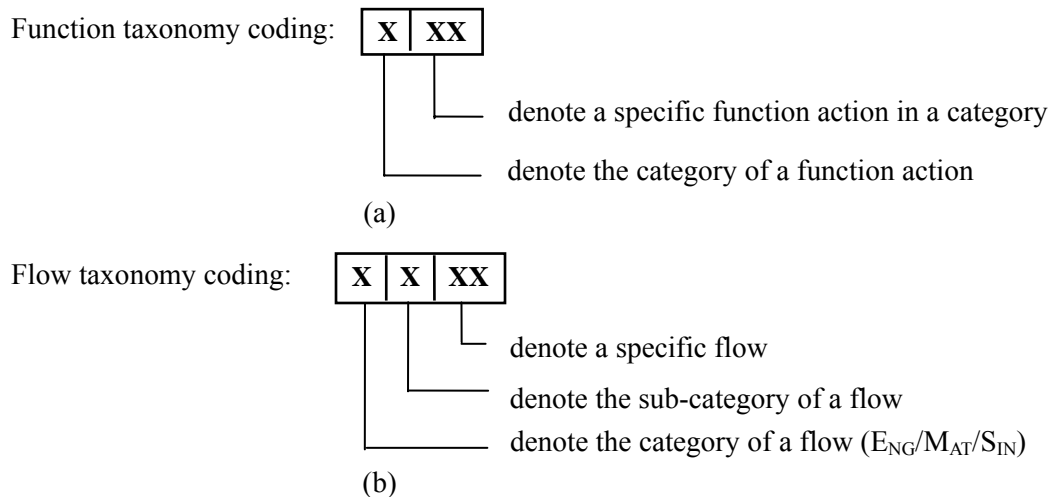


Figure 4.7 Coding schemes of function action and flow taxonomies

Using these coding schemes, each function action and each flow in the taxonomy is assigned a unique code. The function action taxonomy code spans from 000 to 999, and the flow code, 0000 to 9999. The codes are predefined in the system such that a user will use them to build the function models using simple ‘click and assemble’ operations in the graphical user interface (GUI). For example, the code of the ‘heat generation’ function is  $\vec{f} = [302 \quad 1202:3005 \quad 1304]$ .

The taxonomy scheme used in this research can support a broad variety of product functions. However, the taxonomy is not necessarily complete such that some functions may not have been defined in the taxonomy. Two methods are used to solve this problem. Firstly, the system accepts user-defined functions to allow for certain level of ambiguity. Secondly, the taxonomy can be updated to include new vocabularies that are constantly used. Another problem is that the taxonomy scheme is not strictly orthogonal, i.e., the function models are not free of inconsistencies because different designers may use different vocabularies to describe the same function. To alleviate this problem, the function modeling is supported by GUI with ample help text and explanation, such as the pop-up descriptions of a taxonomy and examples to show its usage.

## **4.2 Building Of FPA Using Self-Organizing Map**

The building of product architecture is an indispensable process for organizing the product design information. However, building the product architecture is a

complicated problem because (1) it is difficult to determine the criteria for defining the modularity and commonality of a product architecture, (2) it is a challenging task to determine an appropriate level of granularity for defining product architectures, where granularity refers to the decomposition level of the product information, and (3) a product family usually includes multiple product models which structures may differ in the level of complexities. Therefore, the building of product architecture is a complex process and requires scientific rigor. In essence, the establishment of a product architecture should be consistent with the product information modeling schemes. Function has been advocated by many researchers as a solution-neutral representation of a product (Gero, 1990; Pahl and Beitz, 1996; Gorti and Sriram, 1996; Kirschman and Fadel, 1998; Szykman *et al.*, 1999; Otto and Wood, 2001). Hence, formal representation schemes of product functions can be used to build product architectures. Various methods have been proposed to address this problem, such as the QFD-based approach (Gu and Sosale, 1999; Sand *et al.*, 2002), DSM-based approach (Dong and Whitney, 2001; Yu *et al.*, 2003; Hölttä-Otto, 2005), heuristic approach (Zamirowski and Otto, 1999; Stone *et al.*, 2000b) and quantitative approach (McAdams *et al.*, 1999; Stone *et al.*, 2000a). Despite these efforts, the establishment of a product architecture still relies heavily on human operation and expertise. Computational tools are required to enable rapid and intelligent building of product architecture.

The purpose of FPA is to identify the typical product functions of a product family. In this research, the function analysis operator ( $Op_f$ ) is used to achieve this. An important



step in  $Op_f$  is the clustering of product functions based on the self-organizing map (SOM) method.

#### **4.2.1 Introduction of SOM**

A SOM network is a special class of neural networks based on the theory of competitive learning (Kohonen, 1989). Self organization refers to the evolution of a system into an organized status without external interference. The essence of a SOM is its ability to identify the intrinsic statistical features contained in the input patterns and generate topographic maps (called feature maps) based on unsupervised learning (Haykin, 1999).

A SOM network structure usually consists of three layers, namely, the input layer, the competitive layer and the output layer. Figure 4.8 illustrates a typical SOM model, namely, the Kohonen model (Haykin, 1999). The input layer accepts a multi-dimensional data pattern, which is usually represented as a vector. The competitive layer can be organized into 1- or 2-dimensions. Each neuron receives a summation of the weighted inputs from the input layer, and is associated with a collection of adjacent neurons, which form its 'neighborhood'. Once the network has been initialized, three procedures are involved in the formation of the feature map (Haykin, 1999).

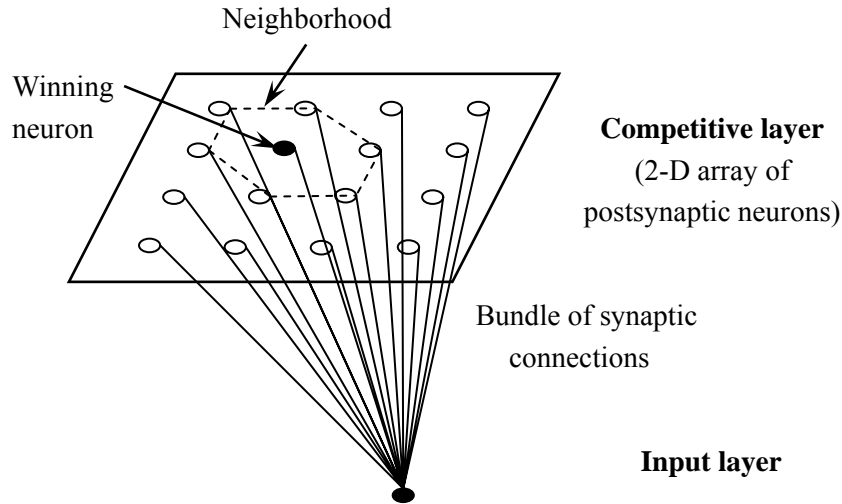


Figure 4.8 Self-organizing map: the Kohonen model\* (Haykin, 1999)

- i. **Competition:** For each input vector, the neurons in the competitive layer compute their responsive values according to a distance function. The neuron with the largest responsive value is declared as the winner. In Figure 4.8, the black neuron is declared as the winner based on a predefined distance function, which is not necessarily the Euclidean distance to the input neuron. However, if the Euclidean distance is used, the winning neuron is usually selected as the one with the minimum distance.
- ii. **Cooperation:** The topological neighbors of the winner are determined to provide the basis for cooperation among them. The winner and its neighbors are collectively called the excited neurons.
- iii. **Synaptic activation:** The excited neurons increase their individual responsive values of the distance function in relation to the input vector.

\* The output layer is not explicitly defined in this model.

This is achieved through adjusting the weight vectors of the excited neurons such that they move towards the input vector.

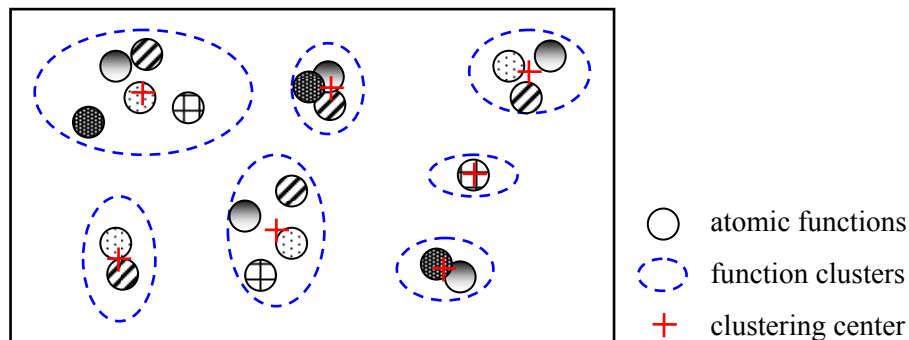
At the initial stage of network formation, no specific order is present. However, after the training processes, the neurons in the competitive layer are self-organized into some meaningful patterns, i.e., the feature map which can arrange the input vectors according to their intrinsic relations. For example, similar input items are clustered close to each other while dissimilar ones are distributed far apart. If the input items are still distributed in the map in a disorder way after the training processes, the feature map is not considered as a meaningful pattern.

The purpose of the output layer is to visualize the interconnections between the nodes in the competitive layer. It does not include any logic for the formation of feature maps, and hence, is not necessary for the proper functioning of a Kohonen network. The output layer is not included in the Kohonen model in Figure 4.8.

#### **4.2.2 Function clustering based on SOM**

SOM is used to perform unsupervised clustering of the product functions. It is expected that some patterns can be identified based on the similarity between functions. The prerequisite of function clustering is a set of products which function structures have been built according to the product information model. Next, all the atomic functions within the function structures belonging to different products are retrieved.

The atomic functions are represented in the form of coded KEVs. The underlying principle is that the attributes contained in the key elements can be considered as coordinates that constitute a multi-dimensional discrete space. Hence, the atomic functions can be viewed as data points distributed in this space. If  $N$  attributes are used to represent the functions, an  $N$ -dimensional space can be constituted. This concept is illustrated with a 2D plane in Figure 4.9. The functions belonging to a set of related products are scattered in this space (the shaded circles). It is expected that similar functions from different products are topographically close to each other and intrinsically fall into a specific group (the ellipse). However, when many products are involved and the structural complexities of these products are diverse, the relationships between the functions are unclear. Thus, these groups are ‘invisible’ to a designer before any computational analysis.



Note: Same shading represents functions belonging to the same product

Figure 4.9 Graphical interpretation of function clustering

A three-layer SOM architecture was built to carry out the function clustering. The input data vector is derived from the KEV. Seven nodes are used to represent seven

elements, namely, *function action*, *input energy flow*, *input material flow*, *input signal flow*, *output energy flow*, *output material flow*, and *output signal flow*. The KEV represents a function using 3-digit or 4-digit codes. However, the SOM neural networks require the input data to be within a range of [0, 1]. Hence, the input data must be normalized. Based on the coding schemes of the function actions and flows, the normalization is carried out in a straightforward way. A function action *xxx* is always in the range of [000, 999]. The normalized code is set as  $xxx/1000$ . Similarly, the normalized flow is set as  $xxxx/10000$ . Table 4.1 shows an example of the input vector for the function ‘heat generation’.

Table 4.1 Input vector of an atomic function – ‘heat generation’

Elements	$A_F$	$I_W(E_{NG})$	$I_W(M_{AT})$	$I_W(S_{IN})$	$O_W(E_{NG})$	$O_W(M_{AT})$	$O_W(S_{IN})$
Value	convert	electric wattage	—	electric signal	heat	—	—
Code	302	1202	0	3005	1304	0	0
Normalized code	0.302	0.1202	0	0.3005	0.1304	0	0

Thus, an input vector can be represented as a vector  $\vec{f}_{i[7 \times 1]}$ , which elements are within the range of [0, 1]. Next, a 2D,  $n$ -by- $n$  lattice is constructed in the competitive layer, where  $n$  is a positive integer depending on the scale of the problem. As a rule of thumb,  $n$  can be set as the average number of atomic functions of the products to be analyzed. Each node in the lattice is connected to the input nodes by a weight vector  $\vec{w}_{j[7 \times 1]}$ . There are a total of  $l=n^2$  weight vectors in the SOM network.

After all the input vectors are imported, a training session is performed according to the three procedures discussed in Section 4.2.1. In the function clustering problem, the following steps are involved.

1. **Initialization:** Random values are assigned to the initial weight vectors  $\bar{w}_j(0)$ .
2. **Matching:** At training step  $p$ , the Euclidean distances between the input vector  $\bar{f}_i$  and the weight vectors  $\bar{w}_j(p)$  are computed. The winning neuron is selected as the one which weight vector has the minimum distance to the input vector.

$$\chi(\bar{f}_i) = \arg \min \| \bar{f}_i - w_j(p) \|, \quad j = 1, 2, \dots, l \quad (4.4)$$

3. **Neighborhood activation:** The neighbors of the winning neuron are selected according to their topographical distances to the winning neuron. For example, the winning neuron itself ( $\chi(\bar{f}_i)$ ) is called  $N(0)$ ; the immediate neighbors of  $\chi(\bar{f}_i)$  are called  $N(1)$ , and so on (Figure 4.10).  $N(i)$  are collectively called the excited neurons.

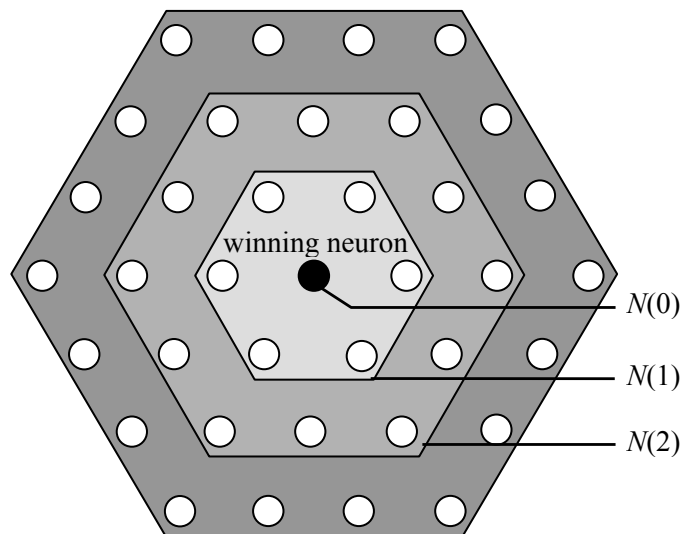


Figure 4.10 Neighborhood activation in a hexagonal lattice

4. **Updating:** The weight vectors  $\bar{w}_j(p)$  of the excited neurons are updated using the following rule.

$$\bar{w}_j(p+1) = \bar{w}_j(p) + \eta(p)h_{j,N(i)}(p)[\bar{f}_i - \bar{w}_j(p)] \quad (4.5)$$

where  $\eta(p)$  is the learning rate, and  $h_{j,N(i)}(p)$  is the neighborhood function, which differs for the excited neurons that locate in different neighborhood  $N(i)$ . Thus, the weight vectors of the excited neurons can be moved slightly towards the input vector. Figure 4.11 illustrates the updating process in a 2D plane.

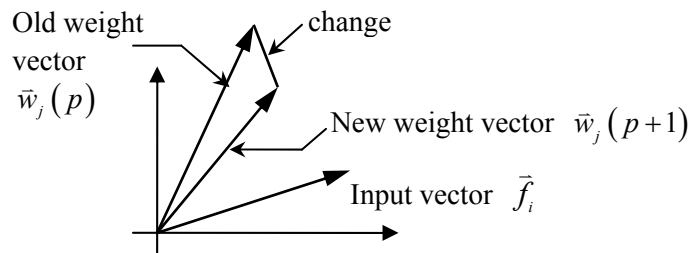


Figure 4.11 Updating weight vector in a 2D plane

5. **Continuation:** The training data is presented to the network repeatedly such that the synaptic weight vectors are updated continuously to resemble the input vectors.

Several trial training sessions can be carried out by varying the controlling parameters, such as the size of the lattice ( $n$ ), the type of lattice (rectangle, hexagonal, random), training epochs, and the learning rates ( $\eta$ ). A visual feature map is generated with the

functions clustered at different nodes.

Finally, a designer is prompted to export the feature map into the output layer, which is organized as a tree structure. A root node represents a cluster entry and the leaf nodes represent the actual functions assigned to this cluster. The tree structure will be further refined by human designers. The refinement process allows the empirical knowledge of the human designers to be incorporated. For example, possible errors that are caused either by insufficient training or by noise data can be identified. Insufficient training happens when the size of the lattice or the training epoch is too small. Noise data emerges when a designer fails to model a product function using suitable vocabularies. The refinement includes operations, such as merging similar functions, deleting non-representative functions, and assigning names to the clusters. The final outcome is a set of common functions for a product family, with the related functions belonging to different product cases assigned to them.

The MATLAB<sup>®</sup> neural networks toolbox provides the basic tools to implement the algorithms of SOM. These tools are adopted in this research in the function analysis program.

### **4.2.3 An illustrative example**

This section presents an illustrative example to build the FPA for a family of electric kettles using the SOM method. The purpose of this example is to show how SOM is



used to cluster product functions and facilitate the building of product architecture. It should be noted that the structure of this product is very simple, and hence, it is not really difficult to build the FPA manually. However, for products which structures are more complicated, or when a large number of products are involved, the SOM method can be of significant advantage.

Four different electric kettles were collected. The function structure of each product was established through function decomposition. Figure 4.12 is the function structure of a particular product. The atomic functions of four different products are shown in Table 4.2. Accordingly, Table 4.3 lists the input vectors that are retrieved from the normalized, coded KEV representation of atomic functions. The coding scheme follows the taxonomies presented in Appendices A and B.

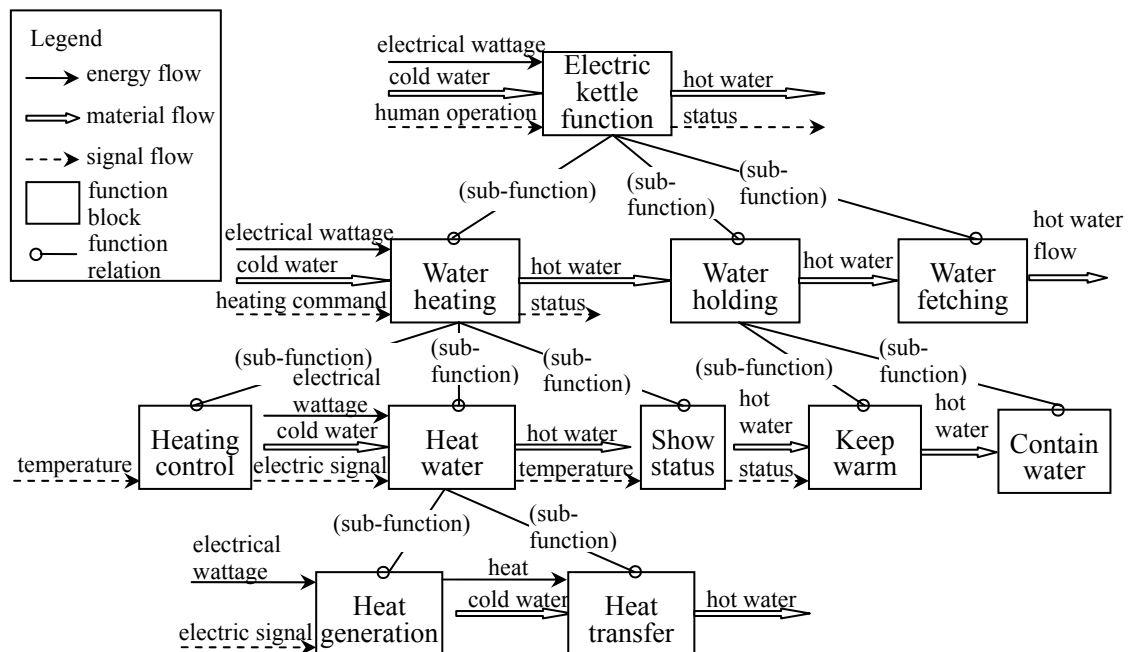


Figure 4.12 Function structure of an electric kettle

Table 4.2 Atomic functions of four sample products

$P_i$	NAME	$A_F$	$I_w-E_{NG}$	$I_w-M_{AT}$	$I_w-S_{IN}$	$O_w-E_{NG}$	$O_w-M_{AT}$	$O_w-S_{IN}$
1	water fetching	convey	human force	water			water	
	heating control	turn-on			temperature			switch on
	show status	display			temperature			visual
	heat generation	convert	wattage		switch on	heat		
	heat transfer	conduct	heat	water			water	
	keep warm	store		water			water	
	contain water	enclose		water				
2	water holding	enclose		water				
	Heating	convert	wattage	water	electric signal		water	
	heating control	turn-on			human operation			switch on
3	heating control	turn-on			human operation			switch on
	generation heat	convert	wattage		electric signal	heat		
	Transfer heat	conduct	heat	solid-liquid			solid-liquid	
	warm keeping	store		solid-liquid			solid-liquid	
	burn protection	constrain	heat			heat		
4	heat control	turn-on			temperature			switch on
	Produce heat	convert	wattage		electric signal	heat		
	Transfer heat	convey	heat	water			water	
	Displace status	display			temperature			visual
	keep warm	store		water			water	
	fetch water	channel	human force	water			water	

Seven nodes were used in the input layer and a 5-by-5 hexagonal lattice was used in the competitive layer. In the lattice, each dot represents a grid node (Figure 4.13), and this node is connected to the input nodes with a weight vector  $\bar{w}_{j[7 \times 1]}$ . Thus, there are 25 weight vectors altogether. The position of a node is denoted as  $N(i, j)$ , where  $i$  and  $j$

are integers, and  $1 \leq i, j \leq 5$ . For example, the origin is denoted as  $N(I, I)$ . Note that this is different from the Cartesian coordinates.

Table 4.3 Normalized input data of the atomic functions

$P_i$	NAME	$A_F$	$I_w-E_{NG}$	$I_w-M_{AT}$	$I_w-S_{NG}$	$O_w-E_{NG}$	$O_w-M_{AT}$	$O_w-S_{IN}$
1	water fetching	0.404	0.1108	0.2102	0	0	0.2102	0
	heating control	0.503	0	0	0.3102	0	0	0.3006
	show status	0.506	0	0	0.3102	0	0	0.3106
	heat generation	0.302	0.1202	0	0.3006	0.1304	0	0
	heat transfer	0.403	0.1304	0.2102	0	0	0.2102	0
	keep warm	0.143	0	0.2102	0	0	0.2102	0
	contain water	0.905	0	0.2102	0	0	0	0
2	water holding	0.905	0	0.2102	0	0	0	0
	Heating	0.302	0.1202	0.2102	0.3005	0	0.2102	0
	heating control	0.503	0	0	0.3	0	0	0.3006
3	heating control	0.503	0	0	0.3	0	0	0.3006
	generation heat	0.302	0.1202	0	0.3005	0.1304	0	0
	Transfer heat	0.403	0.1304	0.2301	0	0	0.2301	0
	warm keeping	0.143	0	0.2301	0	0	0.2301	0
	burn protection	0.902	0.1304	0	0	0.1304	0	0
4	heat control	0.503	0	0	0.3102	0	0	0.3006
	Produce heat	0.302	0.1202	0	0.3005	0.1304	0	0
	Transfer heat	0.404	0.1304	0.2102	0	0	0.2102	0
	displace status	0.506	0	0	0.3102	0	0	0.3106
	keep warm	0.143	0	0.2102	0	0	0.2102	0
	fetch water	0.402	0.1108	0.2102	0	0	0.2102	0

When the input data (the atomic functions) were initially imported, they were distributed in the lattice according to the initial values of the weight vectors (Figure 4.13). The map does not reveal the order of the input vectors. Next, the training process was carried out according to the steps discussed in Section 4.2.2. Experiments were carried out repeatedly with different settings of the learning rates and training epochs. It was found that similar feature maps were built. A typical pattern after the

training process is shown in Figure 4.14.

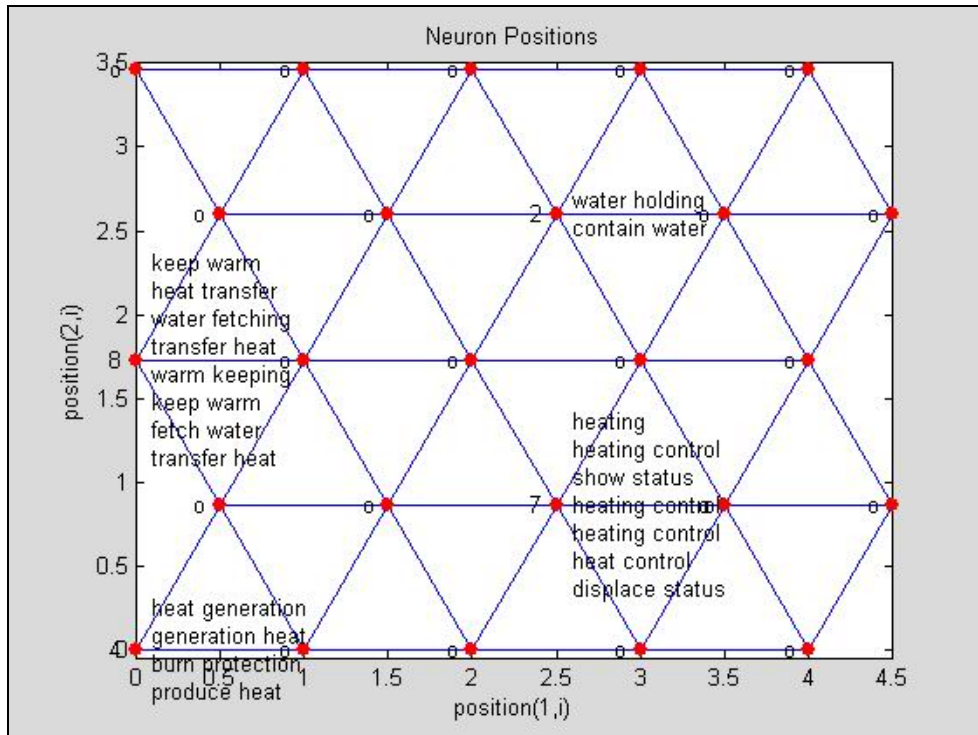


Figure 4.13 Initial status of the competitive layer

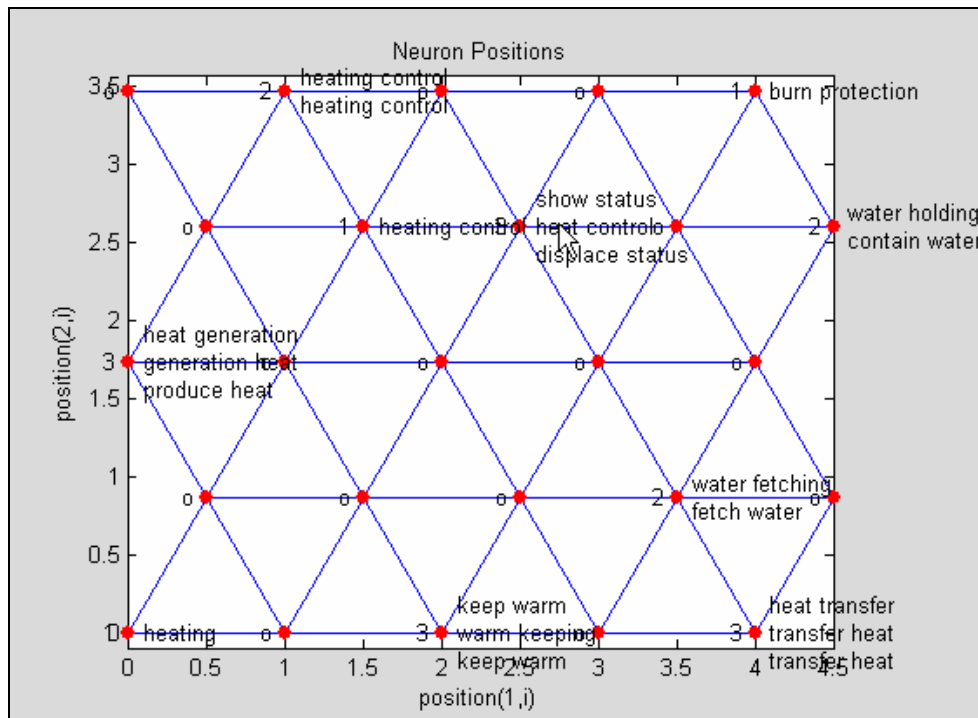
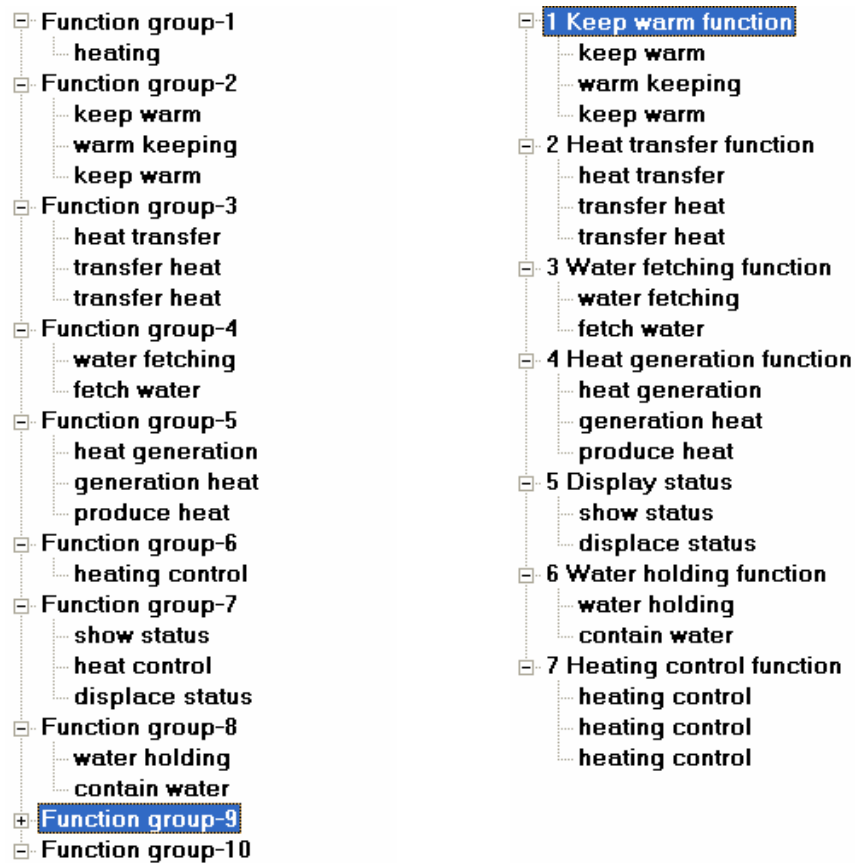


Figure 4.14 Clustering pattern in the competitive layer after training

As can be seen from the feature map, the functions have been grouped according to their affinity to each other. The number on the left of each node represents the number of functions that have been clustered at that node. For example, three functions are located at  $N(I, 3)$ , and they are reasonably considered as a group denoting ‘heat generation’. Based on this pattern, a tree structure was automatically generated as the output layer, where a node of this tree must have at least one function assigned (Figure 4.15(a)). This tree structure was further refined by human designers. For example, the ‘heating’ function located at  $N(I, 1)$  is similar to ‘heat generation’. As a result, it was merged to the ‘heat generation’ cluster located at  $N(I, 3)$ . Finally, seven clusters were identified for the electric kettle products (Figure 4.15(b)). The clusters represent the common functions for a product family, which are denoted as:

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix} = \begin{bmatrix} \text{keep warm} \\ \text{heat transfer} \\ \text{water fetching} \\ \text{heat generation} \\ \text{display status} \\ \text{water holding} \\ \text{heating control} \end{bmatrix} .$$



(a) Initial clustering based on SOM

(b) Architecture after manual refinement

Figure 4.15 FPA of the electric kettle products

#### 4.2.4 Evaluation of the SOM method

The formation of the feature map requires the function base of the product information model. Function similarity is the underlying rule to cluster the functions, where function similarity is estimated based on the coded functions and flows. These are analogous to the quantitative and heuristic methods used to build modular architectures (McAdams *et al.*, 1999; Zamirowski and Otto, 1999; Stone *et al.*, 2000a, 2000b). The resulting function clusters are similar to the product platforms obtained in these methods. However, the SOM method adopts a process that is different from these methods. In particular, unsupervised learning is used for function analysis.

SOM is a necessary step although a refinement process can be carried out subsequently by a human designer. The reasons are, firstly, the SOM method analyzes the data based on the KEV while a designer relies more on empirical information; secondly, a manual analysis of the data from the KEV will involve significant effort, especially when a large number of products are included and the product function structures are complicated; and thirdly, the designer may not be aware of the appropriate numbers or patterns of the clusters at the beginning of the function analysis. Therefore, the advantages of using the SOM method are: (1) an expedition of the process, (2) an alleviation of human labor, (3) determination of useful initial knowledge and patterns of the architecture, and (4) an analysis of the data from a perspective other than empirical observation. Therefore, the SOM process and human refinement are complementary to each other.

To summarize, the SOM method can generate a feature map from a set of functions without human supervision. Thus, it can identify the preliminary patterns of the functions and facilitate the building of product architecture. This method is different from product platform design using the top-down approaches which require tremendous product analysis (Simpson *et al.*, 2001). In comparison with the approaches to build product architecture in modular design (McAdams *et al.*, 1999; Zamirowski and Otto, 1999; Stone *et al.*, 2000a, 2000b), the SOM method is a computational technique using unsupervised learning algorithms. Therefore, SOM has less reliance on human expertise. Although the final formation of the product

architecture still requires refinement by human designers, the refinement process is carried out after a feature map has been formed such that a rudimentary architecture is already present. Therefore, the refinement process will not pose a heavy load on a user.

### **4.3 Establishment Of Product Platform**

Typically, design involves the process of finding the proper design parameters to fulfill the design requirements. Therefore, it is important to: (1) identify the design parameters, (2) identify relevant design requirements with suitable measurements, and thereafter, (3) establish the relationships between the design requirements and the design parameters. Such relationships can be broadly considered as a mapping route from the design requirements to the design parameters.

To achieve these, a few knowledge extraction operators  $\{Op_x\}$  are developed. The overall process is organized as a domain mapping process similar to the zig-zag decomposition in axiomatic design (Suh, 2001) (Figure 4.16). Among these operators,  $Op_f$  has been defined to establish the FPA.  $Op_k$  and  $Op_r$  are discussed in this section, and  $Op_i$  and  $Op_c$  will be discussed in the subsequent chapters.

The top-level design requirements are the customer requirements (CRs), which include the most important product features that define the market segmentations. Market analysis can be carried out to obtain the set of CRs. Since this research focuses on the



engineering aspects, it is assumed that the CRs are known, and are denoted as  $\mathbf{r} = [r_1, r_2, \dots, r_s]^T$ . For example, the customer requirements of the electric kettle are:

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} = \begin{bmatrix} \text{energy usage} \\ \text{capacity} \\ \text{safety} \\ \text{ease of use} \\ \text{cost} \end{bmatrix}.$$

The CRs are quantified by KCs. The extraction of a set of KCs to signify the performance features and define a product family is discussed in Section 4.3.1. The KCs are dependent on a set of functions, which is defined in the FPA. The bottom-level design parameters are the physical components, which are contained in the component catalog. The formation of the component catalog is discussed in Section 4.3.2.

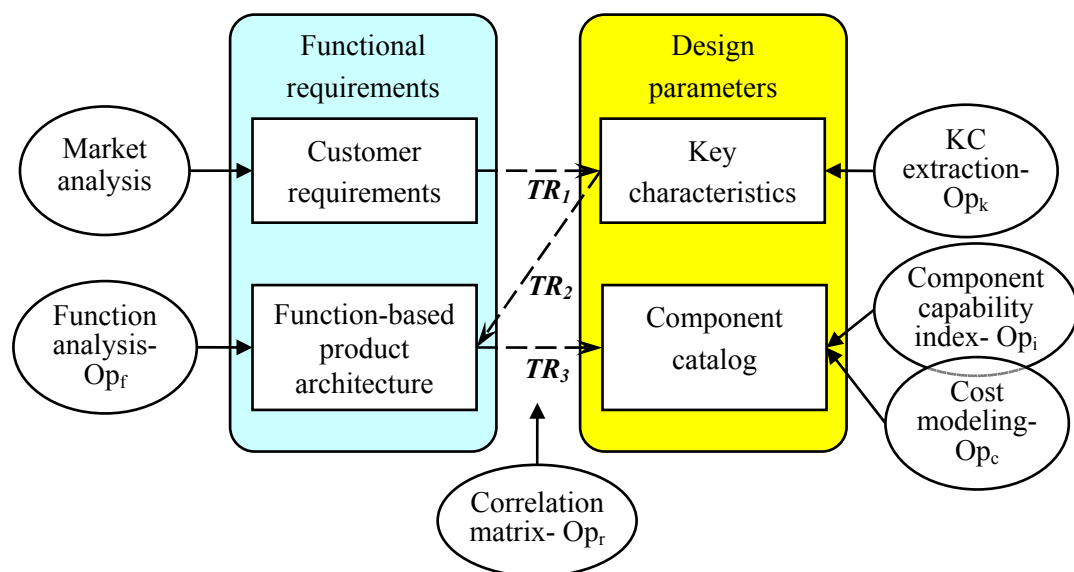


Figure 4.16 Mapping route from design requirements to design parameters

### 4.3.1 Extraction of KCs as performance criteria

As mentioned earlier, each product ( $p_i$ ) contains a set of KCs:  $K_i^0 = [k_{i1}, k_{i2}, \dots, k_{ip}]^T$ .

The set of KCs may be different across different products. However, since the products collected in the same general design space (SP<sub>i</sub>) are similar to each other and are expected to form a product family, it is possible to identify a set of common KCs to measure the performance of the products. This process is carried out manually. The resulting set of common KCs is denoted as  $\mathbf{k} = [k_1, k_2, \dots, k_p]^T$ . As an example, the KCs for the electric kettle product family include:

$$\mathbf{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \\ k_7 \end{bmatrix} = \begin{bmatrix} \text{power consumption} \\ \text{dimension} \\ \text{water capacity} \\ \text{automatic control} \\ \text{cost} \\ \text{MTBF}^* \\ \text{water fetching method} \end{bmatrix}.$$

\* MTBF: Mean Time Between Failures

In the subsequent design synthesis and evaluation stage,  $\mathbf{k}$  is used as the major performance criterion that differentiates the product members in the product family.

### 4.3.2 Formation of component catalog

Each product ( $p_i$ ) contains a set of physical components:  $M_i^0 = [m_{i1}, m_{i2}, \dots, m_{in}]^T$ .

Components that belong to different products can be collected into the component catalog and will be used as the design parameters in the design synthesis and

evaluation stage. The component catalog is organized as a set of slots that correspond to the common functions defined by the FPA. Components that implement the same product function are assigned to the same slot. Thus, the relationships between the components and the functions are retained. The components are collectively denoted as:

$$\mathbf{m} = [s_1, s_2, \dots, s_m]^T$$

where  $s_i$  is the  $i^{\text{th}}$  slot which corresponds to the common function  $f_i$ , and

$$s_i = [m_i^1, m_i^2, \dots, m_i^{n_i}]^T$$

where  $n_i$  is the number of components in this component slot.

Other than the attributes that characterize each component, the component cost and the performance capability should also be identified. Cost can be identified based on historical data. A cost road-map is established for each component. Performance capability refers to the capability of a component to satisfy specific design requirements. It is extracted using the component capability index operator ( $Op_i$ ), as will be discussed in Chapter 5.

### 4.3.3 Establishment of mapping route using correlation matrices

Based on the above discussions, the design parameters and design requirements at different levels have been defined. Given that  $\mathbf{r}_{s \times 1}$ ,  $\mathbf{k}_{p \times 1}$ ,  $\mathbf{f}_{m \times 1}$ ,  $\mathbf{m}_{n \times 1}$  represent the vectors of CRs, KCs, common functions, and physical components in the component catalog,

respectively, a mapping route can be established using a few correlation matrices, which constitute the  $Op_r$  process.

$$r = TR_1 \times k \quad (4.6)$$

$$k = TR_2 \times f \quad (4.7)$$

$$f = TR_3 \times m \quad (4.8)$$

$TR_1$  is an  $s \times p$  matrix which element  $(i, j)$  is either ‘1’ indicating  $r_i$  is related to  $k_j$ , or ‘0’ indicating  $r_i$  is not related to  $k_j$ . A similar definition is made for  $TR_2$  and  $TR_3$ .  $TR_1$  and  $TR_2$  can be designed manually, similar to the QFD processes (Martin and Ishii, 2002). In addition,  $TR_2$  can be assisted by the  $X_{K-F}$  relationship defined in the contextual facet of the product information model.  $TR_3$  is a natural outcome of the establishment of the component catalog. Using the electric kettle as an example,  $r$ ,  $k$ , and  $f$ , have been defined in previous sections.  $TR_1$  and  $TR_2$  are established as shown in Table 4.4 and Table 4.5, respectively. The overall mapping route is illustrated in Figure 4.17.

Table 4.4 Correlation between CRs and KCs ( $TR_1$ )

$TR_1$		$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$
$r_1$	Energy usage	1	0	0	0	1	0	0
$r_2$	Capacity	0	1	1	0	1	0	0
$r_3$	Safety	0	0	0	1	1	1	0
$r_4$	Ease of use	0	0	0	1	0	0	1
$r_5$	Cost	0	0	0	0	1	0	0
$k_1$ : power consumption		$k_2$ : dimension		$k_3$ : water capacity		$k_4$ : automatic control		
$k_5$ : cost		$k_6$ : MTBF		$k_7$ : water fetching method				

‘1’ indicates that a CR is dependent on a KC; ‘0’ otherwise.

Table 4.5 Correlation between KCs and functions ( $TR_2$ )

KCs		$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$k_1$	power consumption	0	1	0	1	0	0	0
$k_2$	dimension	0	0	0	0	0	1	0
$k_3$	Water capacity	0	0	0	0	0	1	0
$k_4$	automatic control	1	0	0	0	0	0	1
$k_5$	Cost	1	1	1	1	1	1	1
$k_6$	MTBF	1	1	1	1	1	1	1
$k_7$	Water fetching	0	0	1	0	0	0	0

$f_1$ : Keep warm     $f_2$ : Heat transfer     $f_3$ : Water fetching     $f_4$ : Heat generation  
 $f_5$ : Display status     $f_6$ : Water holding     $f_7$ : Heating control

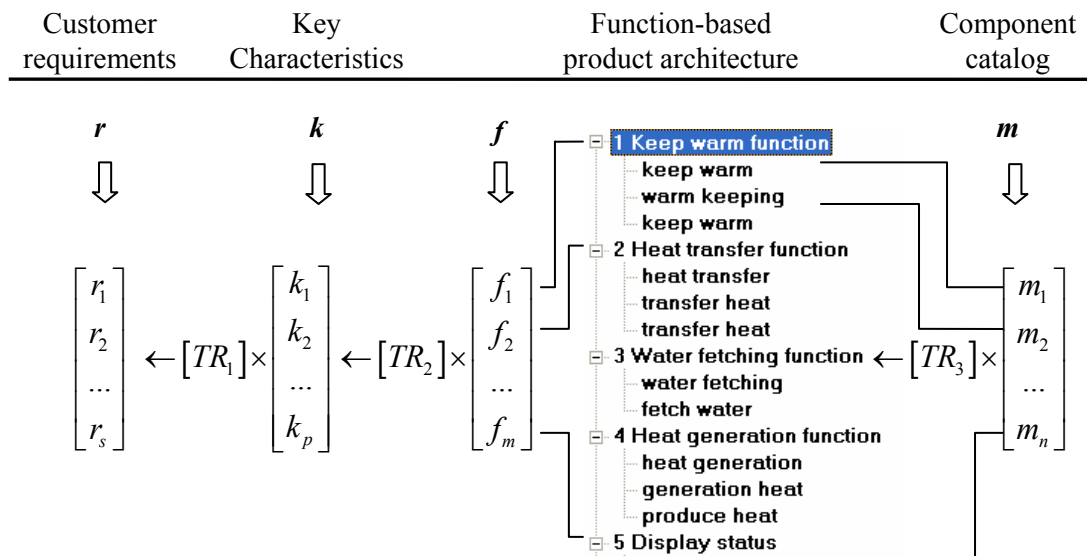


Figure 4.17 Mapping route from CRs to physical components

The mapping route from the design requirements to the design parameters is similar to the domain mapping process in axiomatic design (Suh, 2001). However, there are a few differences between the two methods.

Firstly, while the concept of domain mapping is common in both methods, axiomatic

design is not targeted at deriving the contents in each domain, such as functional requirements (FRs) and design parameters. In comparison, the design reuse method explicitly defines the steps to derive the contents (e.g., KCs, functions, and components) based on the knowledge extraction operator.

Secondly, the axiomatic design (Suh, 2001) requires the FRs be independent of each other (the independence axiom). The independence axiom is not mandatory in this research because the KCs in this research are not equivalent to the FRs in axiomatic design. Strict independence between KCs may be difficult to be achieved, especially for variant design (Jiao and Tseng, 1998). Hence, the design reuse method was developed without the restriction of uncoupled or decoupled design (Suh, 2001). While this may create some potential problems concerning the product performance evaluation, it is inevitable from a design reuse perspective.

Thirdly, the domain mapping process proposed in axiomatic design provides general guidelines to design problem solving, such that the mapping scheme between different domains may appear in different forms, such as DSM (Dong and Whitney, 2001) and association rules (Jiao and Zhang, 2005). In this research, the mapping route defined in the product platform is in the form of correlation matrices. In such a sense, the axiomatic design is a more generic method. The correlation matrices used for the establishment of product platform are a special application of the domain mapping process in axiomatic design.

#### 4.4 SUMMARY

A function-based product information model is used to model existing product cases. A set of knowledge extraction operators  $\{Op_x\}$  is defined to establish a product platform, thus, transforming the raw data into reusable forms. In particular, the function analysis operator ( $Op_f$ ) is used to cluster product functions, leading to the FPA. SOM is a new tool to facilitate the function clustering based on unsupervised learning techniques. The SOM method is advantageous as compared to traditional methods, such as manual methods and heuristic methods, in that it has less reliance on human experience and can alleviate human efforts. A mapping route is established between the design requirements and the design parameters using several correlation matrices. A component catalog is used to contain the physical components. These establish the knowledge base to support subsequent design synthesis and evaluation.

# **Chapter 5 ICA METHOD FOR PRODUCT PERFORMANCE EVALUATION**

This chapter focuses on the evaluation of product performance. An information content assessment (ICA) method is proposed to evaluate product performance based on the principles of the information axiom (Suh, 2001). This method involves systematic steps to identify the component capability index at the knowledge extraction stage, and the calculation of the product information content at the design synthesis and evaluation stage. Finally, a few precautions are discussed for the application of this method.

## **5.1 Product Performance Evaluation**

The underlying principle of design reuse is design-by-analogy. Solutions are generated from existing components, which quality has been proven in designed products. In such a sense, the quality of the products can be better predicted and ensured. Therefore, it is reasonable to evaluate the performance of a new product based on existing similar designs. Considering the rapid advances in technology, only elements of products belonging to the existing technology are reused in new designs. Given that a performance criterion is influenced by one or a few components (i.e., the design parameters), the outcome can be predicted from similar designs that make use of



identical components.

To effectively carry out performance evaluation, two issues are important: (1) how to establish the relationships between the design parameters and the performance criteria, and (2) how to deal with the multiple performance criteria that are inherently incommensurable. The first issue has been discussed in Chapter 4. For the second issue, information content is proposed as a uniform metric of the multiple performance criteria.

## **5.2 The Information Content Assessment (ICA) Method**

### **5.2.1 Background**

Information content is defined in terms of the probability that the FRs can be satisfied by the design parameters. According to the information axiom, among the solutions that satisfy the independence axiom, the solution with the minimum information content is the optimal one (Suh, 2001). In addition, information content can be used as a measurement of the complexity of a design, i.e., the uncertainty in achieving the specific FRs (Suh, 2005). Thus, given the same set of FRs, a design with a lower information content involves less complexity than the one with a higher information content, and thus is superior to the latter.

To calculate the information content with respect to a specific design requirement, first, the achievable performance must be computed from the design parameters, as they are

set in the designed products. The achievable performance is called the system range, which can be generalized as a probability density function (pdf). Figure 5.1 uses a normal distribution to represent the system range. At the same time, the customers set the design requirements, which can be quantified as the design ranges, usually in the form of the lower and upper bounds of the target product performance. Next, the system range is compared with the design range, resulting in a common range. Thus, a probability ( $\zeta_i$ ) is obtained to show how well the design requirement can be satisfied by the design parameters. Information content ( $I_i$ ) is computed as the logarithmic function of this probability (Suh, 2001), as,

$$I_i = \log_2 \frac{1}{\zeta_i} = -\log_2 \zeta_i \quad (5.1)$$

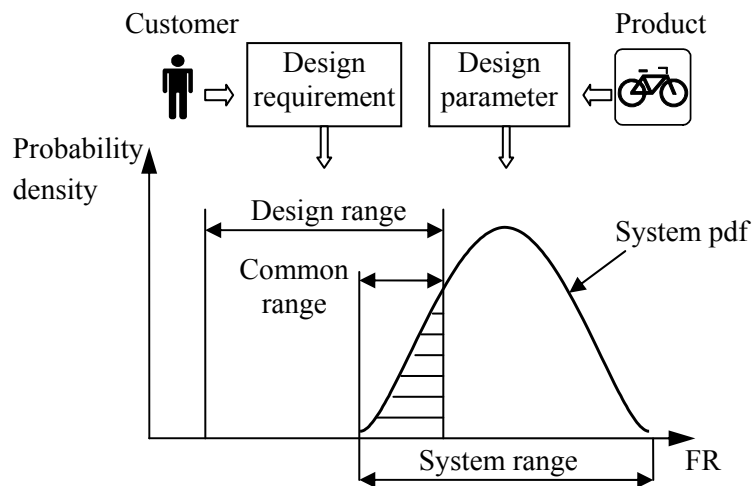


Figure 5.1 Relationship between design range and system range

In the case of a design task with  $p$  design requirements, the information content of the entire system ( $I_{sys}$ ) is determined using,

$$I_{sys} = -\log_2 \zeta_{\{p\}} \quad (5.2)$$

where  $\zeta_{\{p\}}$  is the joint probability that all requirements are satisfied.

Provided that the design requirements are independent of each other,  $\zeta_{\{p\}}$  can be computed as the multiplication of the individual probabilities ( $\zeta_i$ ), i.e.,  $\zeta_{\{p\}} = \prod_{i=1}^p \zeta_i$ .

Thus, the information content of the system can be computed using,

$$I_{sys} = -\log_2 \prod_{i=1}^p \zeta_i = -\sum_{i=1}^p \log_2 \zeta_i \quad (5.3)$$

Equation (5.3) indicates that for a system which design requirements are independent of each other, the information content of the system is the summation of individual information content, i.e.,  $I_{sys} = \sum_{i=1}^p I_i$ . However, if the design requirements are not independent, the information content of a product cannot be computed this way. The correlations between the design requirements have to be established to compute the joint probability that the design requirements can be satisfied. Considering the difficulties to establish the correlations between the design requirements, it may involve tremendous efforts to compute the information content. Therefore, in this research, without the restriction of the independence of the design requirements, the summation of the individual information content is used as a measure of the overall product performance, which can be considered as the pseudo information content. For description consistency, the term ‘information content’ is used to refer to the pseudo information content with respect to a product, as well as the information content with

respect to individual design requirements.

An attractive feature of using the information content in performance evaluation is that it is a dimensionless measurement based on probability. Hence, it provides a uniform metric to incorporate various measures of technical criteria that are inherently incommensurable (Jiao and Tseng, 1998).

To effectively use the information content for performance evaluation, it is required that the system range and design range of a product be established. However, the establishment of the system range has been difficult and lacked effective computational tools. Therefore, this research proposes the ICA method to establish the system ranges and assess the information content.

### **5.2.2 Procedures of the ICA method**

The ICA method involves six steps as shown in Figure 5.2. Each step is supported by relevant tools or information resources (shown in the ellipsis). Steps 1 ~ 3 are carried out in Stage II of the PFDR process model. They are responsible for establishing the component capability indices, which define the system ranges of the physical components. Steps 4 ~ 6 are carried out at the design synthesis and evaluation stage (Stage III), during which the configurations of products are generated and the information content is calculated. The steps are discussed in Sections 5.2.3 and 5.2.4, using the electric kettle product as an illustrative example.

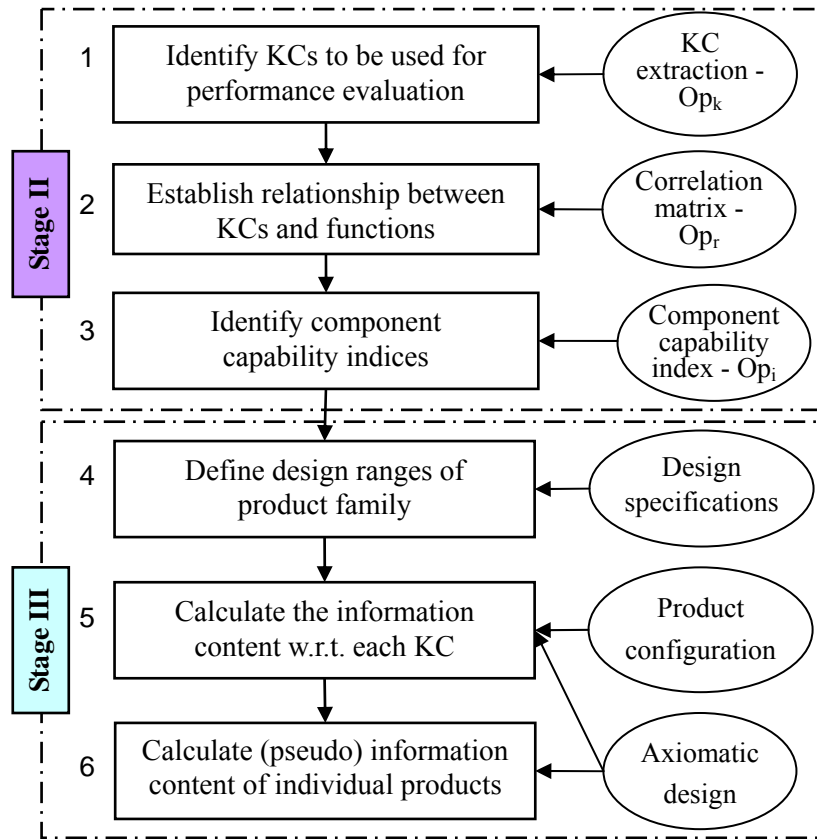


Figure 5.2 The processes of the ICA method

### 5.2.3 Establishment of system range from existing products

Product performance is evaluated against the KCs. Therefore, the relevant KCs have to be identified first. Moreover, the correlation matrices,  $TR_2$  and  $TR_3$ , are required to establish the mapping route from the KCs to the physical components in the component catalog. Steps 1 and 2 are designed to carry out these processes, supported by the operators  $Op_k$  and  $Op_r$ , respectively. These issues have been discussed in Chapter 4 and the results for the electric kettle product are reproduced in Table 5.1 and Table 5.2, respectively. In addition, Table 5.3 presents the component slots that correspond to the common product functions. The number of components within each slot is listed in the last column.

Table 5.1 KCs of the electric kettle

KCs		Type	Unit	Default value
$k_1$	Power consumption	real	watt	–
$k_2$	Dimension	real	mm	–
$k_3$	Water capacity	real	liter	–
$k_4$	Automatic control	Boolean	–	True(1), False(0)
$k_5$	Cost*		S\$	
$k_6$	MTBF	real	hour	–
$k_7$	Water fetching method	categorical	–	manual, tap, air pressure

Table 5.2 Correlation between KCs and functions ( $TR_2$ )

KCs	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$k_1$	0	1	0	1	0	0	0
$k_2$	0	0	0	0	0	1	0
$k_3$	0	0	0	0	0	1	0
$k_4$	1	0	0	0	0	0	1
$k_5$	1	1	1	1	1	1	1
$k_6$	1	1	1	1	1	1	1
$k_7$	0	0	1	0	0	0	0

$f_1$ : Keep warm     $f_2$ : Heat transfer     $f_3$ : Water fetching:     $f_4$ : Heat generation  
 $f_5$ : Display status     $f_6$ : Water holding     $f_7$ : Heating control

Table 5.3 Function and component slot

Function		Component slot		Number of components
$f_1$	Keep warm	$s_1$	Insulator	4
$f_2$	Heat transfer	$s_2$	Contact disk / radiator	3
$f_3$	Water fetching	$s_3$	Outlet device	3
$f_4$	Heat generation	$s_4$	Heating disk / coil	7
$f_5$	Display status	$s_5$	LED / LCD screen	5
$f_6$	Water holding	$s_6$	Container	6
$f_7$	Heating control	$s_7$	Thermometer and switch	3

The key step in the ICA method is Step 3, through which the capability indices of the

\* Cost is not considered in performance evaluation because it is not a performance factor.

individual components (if a KC is related to one component) or component combinations (if a KC is related to more than one component) are extracted. The processes are illustrated in Figure 5.3 and explained using an example of a component combination of the electric kettle product.

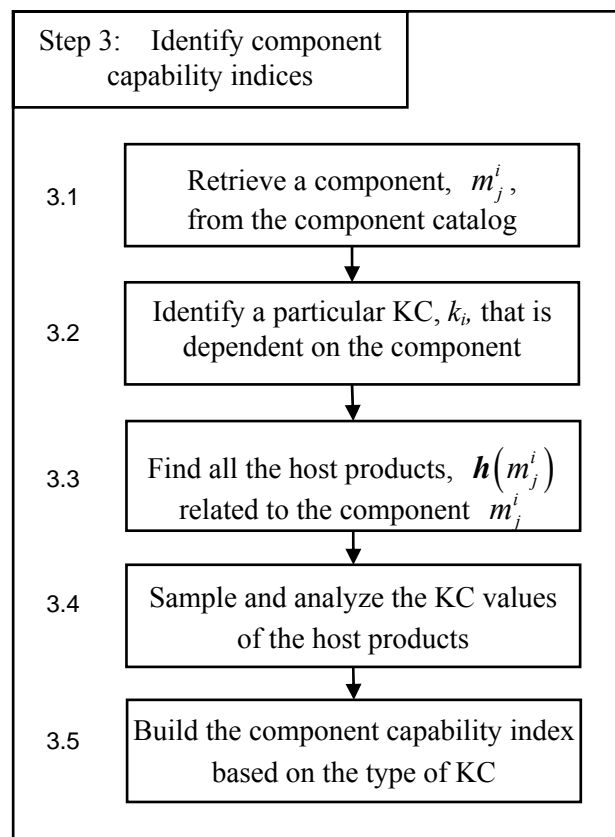


Figure 5.3 Computing the component capability indices

### Step 3.1

Retrieve a component or a component combination from the component catalog, which is represented as  $(m_j^i)$ , where  $j$  is the index of the component slot ( $s_j$ ) and  $i$  is the index of the component in slot  $s_j$ . For example, the component combination,  $(m_2^1, m_4^1) = (3\text{-layer radiator, heating disk})$  of the electric kettle product is retrieved.

The capability index of  $(m_2^1, m_4^1)$  is to be established.

### Step 3.2

The component is related to a particular product function  $f_i$ , through the correlation matrix  $TR_3$ , which in turn, is related to a particular KC,  $k_i$ , through the correlation matrix  $TR_2$ . Following the same example, the component combination  $(m_2^1, m_4^1)$  influences functions  $(f_2, f_4) = (\text{heat transfer}, \text{heat generation})$ , as is derivable from  $TR_3$ . Functions  $(f_2, f_4)$  determine  $k_1 = \text{power consumption}$ , as is shown by  $TR_2$  (Table 5.2).

### Step 3.3

All the host products are retrieved for component  $(m_j^i)$ . In this research, the product cases from which a component are extracted are called the **host products**, and are denoted as  $\mathbf{h}(m_j^i) = [h_j^1, h_j^2, \dots, h_j^u]$ , where  $u$  is the number of products that host the particular component. Since the same component may be used in different models of the products, each distinct component or component combination may be related to a number of host products. In this case study, the component combination  $(m_2^1, m_4^1)$  is used in three host products, namely,  $\mathbf{h}(m_2^1, m_4^1) = [h^1, h^2, h^3]$  as is shown in Table 5.4.

### Step 3.4

With respect to  $k_i$ , the KC value of each host product is obtained. The KC values



can be discrete or continuous depending on the type of KC. For the continuous type of KC, the KC values usually vary in a certain range. Several samples can be retrieved for each host product, and their KC values are obtained. For example, with respect to  $k_1$ : *power consumption*, ten samples are retrieved for every product  $h^i$  ( $i=1, 2, 3$ ) (Table 5.4). The *power consumption* value of these samples can be identified based on historical data.

Table 5.4 Sampled *power consumption* values of three products that host  $(m_2^1, m_4^1)$

Sample	$h^1$	$H^2$	$h^3$
1	1461	1490	1517
2	1475	1505	1516
3	1471	1484	1506
4	1475	1514	1509
5	1474	1519	1512
6	1477	1514	1534
7	1512	1531	1540
8	1480	1516	1528
9	1447	1499	1498
10	1496	1498	1499

### Step 3.5

The KC values related to the sampled host products are merged to form a distribution function, known as the capability index of a component. This can be done manually if the distribution is simple. Alternatively, for a more complex data pattern, the distribution function can be built using statistical methods, such as the parametric estimation (Siddall, 1983). Thus, the capability index can be established, in the form of a pdf for a continuous type of KC, or a probability mass function

(pmf) for a discrete type of KC. Using the same example, the pdf for the component combination  $(m_2^1, m_4^1)$  can be formulated as a normal distribution based on the sample data in Table 5.4. The capability index is defined as  $\{(m_2^1, m_4^1) | k_1 : \mu = 1499, \sigma = 20.6\}$  and is plotted in Figure 5.4.

A few typical distribution functions for representing the capability indices with respect to different types of KCs are shown in Figure 5.5.

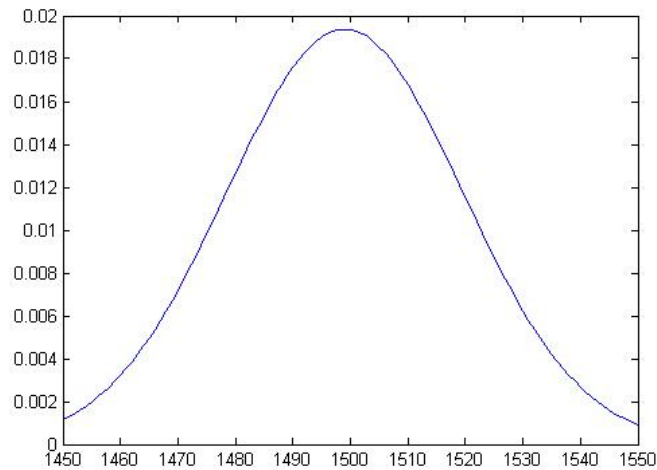


Figure 5.4 Capability index for component combination  $(m_2^1, m_4^1)$

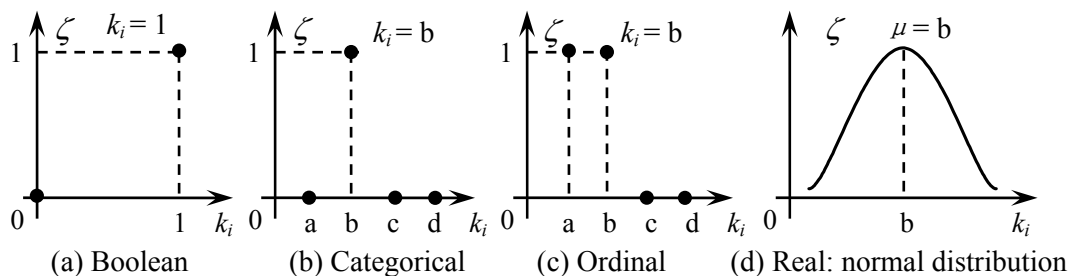


Figure 5.5 Typical capability indices for different types of KCs

In general, the capability indices of the components can be established using the above steps. However, there is one situation where the above steps are not effective. This situation arises when a KC is related to a number of functions, which in turn are related to a number of physical components, such that there could be too few component combinations available from existing product cases. In such a situation, it is preferable to identify the exact relationships between the attributes of the components and the KCs. Such relationships can be in the form of equations or meta-models (Simpson, 1998). Accordingly, the system range can be computed as follows.

Assuming that the product performance is denoted as  $k$ , and the related component attributes are denoted as a vector  $\mathbf{t}=[t_1, t_2, \dots, t_q]$ , an equation can be established as  $k=\psi(\mathbf{t})$ , where  $\psi$  is the mapping function. Given that the distribution of the design parameter  $\mathbf{t}$  is known, the distribution of the product performance  $k$  can be estimated using the first order Taylor series expansion of the function (Nayak *et al.*, 2002). For example, if the mean and variance of the design parameters  $\mathbf{t}$  are  $\mu_{t_i}$ , and  $\sigma_{t_i}^2$  ( $i=1, 2, \dots, q$ ), the mean and variance of the performance  $k$  can be computed as:

$$\mu_k = \psi(\mu_{t_1}, \mu_{t_2}, \dots, \mu_{t_q})$$

$$\sigma_k^2 = \sum_{i=1}^q \left( \frac{\partial \psi}{\partial t_i} \right)^2 \sigma_{t_i}^2$$

Thus, the distribution function that defines the system range can be established. This is an alternative to the correlation matrix-based method to establish the capability

indices.

#### 5.2.4 Calculation of information content

Typically in product family design, a ranged set of design requirements is initiated to cover the target market segmentations. This corresponds to Step 4 of the ICA method. The design requirements are interpreted as a set of design ranges according to the KCs. For example, the designer may wish to launch a set of three models of electric kettles, which design requirements are defined in Table 5.5.

Table 5.5 Design requirements of a family of electric kettle products

KCs	$P_1$	$P_2$	$P_3$
Power consumption (W)	$800 \pm 5\%$	$1000 \pm 5\%$	$1100 \pm 5\%$
Water capacity (L)	1.7~1.8	2.5~2.55	2.4~2.5
Automatic control	True	True	True
MTBF (h)	$\geq 20,000$	$\geq 18,000$	$\geq 26,000$
Water fetching method	manual	air pressure	tap

In Step 5, the design synthesis algorithms generate the product configurations as combinations of physical components. From the product configurations, the component or component combination that determines each KC can be easily identified. Next, the system range can be obtained from the capability indices that have been determined in Step 3. At the same time, the design range has been determined in Step 4. The information content with respect to this KC can be calculated from the relationship between the system range and the design range (refer to Figure 5.1). For example, given that the design range of a continuous type KC ( $k_i$ ) is  $[dn\_L, dn\_U]$ , and

the pdf of the system range is  $\zeta(k_i)$ , the information content can be computed as:

$$I(k_i) = -\log_2 \left( \int_{dn-L}^{dn-U} \zeta(k_i) dk_i \right) \quad (5.4)$$

Equation (5.4) is a special case of Equation (5.1), i.e., when the respective KC has a continuous value and the system range pdf is known. For the electric kettle example, the configuration for  $P_I$  is presented in Table 5.6. The information content with respect to the KCs is computed in Table 5.7.

Finally, in Step 6, the pseudo information content of a product is computed. Based on Equation (5.3), the pseudo information content of electric kettle  $P_I$  is:

$$I'(P_I) = \sum_{i=1}^5 I(k_i) = 0.11 + 0 + 0 + 0.03 + 0 = 0.14.$$

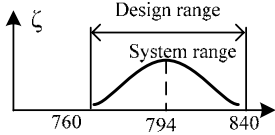
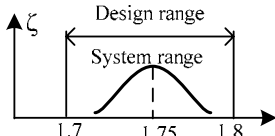
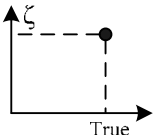
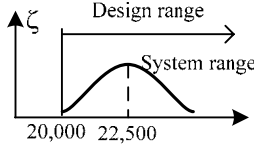
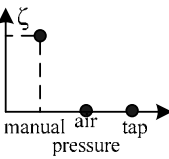
Information content is a measurement of the probability that the design requirements can be satisfied based on the available capability of the designed cases. Using the reverse form of Equation (5.1), the probability that the design requirements can be satisfied can be computed from the information content:

$$\zeta_i = 2^{-I_i} \quad (5.5)$$

Table 5.6 Product configurations of an electric kettle

Component slot		Component used in $P_I$	
$s_1$	Insulation	$m_1$	nil
$s_2$	Contact disk / radiator	$m_2$	Copper radiator
$s_3$	Outlet device	$m_3$	Tap outlet
$s_4$	Heating disk / coil	$m_4$	Resistance coil 3.5A
$s_5$	LED / screen	$m_5$	Boiling LED
$s_6$	Container	$m_6$	1.75L Plastic container
$s_7$	Thermometer and switch	$m_7$	Auto power cut

Table 5.7 Computation of information content

KC	Design range	Component	System range	Probability	Information content
Power consumption	[760, 840]	$(m_2, m_4)$	$\mu=794$ $\sigma=21.5$	$\zeta=0.9269$ 	0.11
Water capacity	[1.7, 1.8]	$(m_6)$	$\mu=1.75$ $\sigma=0.015$	$\zeta=0.9991$ 	0
Automatic control	True	$(m_1, m_7)$	True	$\zeta=1$ 	0
MTBF	$\geq 20,000$	$(m_4)$	$\mu=22,500$ $\sigma=1,255$	$\zeta=0.9768$ 	0.03
Water fetching method	manual	$(m_3)$	manual	$\zeta=1$ 	0

The information content of the electric kettle product  $P_I$  is 0.14, which means that the probability that all the design requirements can be satisfied was approximately 90%. This gives a designer an idea of the overall performance of the product. The designer can focus on one or a few KCs which information content is significantly larger than the others. The components related to those KCs can thus be modified to enhance the relevant product performance.

Information content integrates various product performance criteria into a scalar value, which makes it convenient to compare different products. In addition, the design synthesis problem can be formulated into an optimization problem that allows for easy computational support. This topic will be discussed in Chapter 6.

### **5.2.5 A comparison of the ICA method and axiomatic design**

Axiomatic design provides the theoretical foundation for the ICA method. Hence, there are a number of similarities between these two methods. However, these two methods are proposed at different levels of abstraction. Being a generic method, axiomatic design does not address the issue of how to extract the system ranges. System ranges are either assumed to be known (Bahrami, 1994; Jiao and Tseng, 1998; 2004), or are constructed based on extensive computations and detailed analyses in specific problem contexts (Suh, 2001). Moreover, the FRs are not necessarily quantifiable, making the computation of the information content difficult, if not impossible. On the other hand, the ICA method is proposed within the framework of design reuse. Based on this

framework, improvements have been made in the following two aspects. Firstly, the KCs are quantifiable and are well-understood engineering measurements. In particular, four types of KCs have been defined to cover the general engineering specifications. The definition of KCs is generic, and hence is independent of the product domains. This has made it easier to formulate the system range and design range, and to compute the information content. Secondly, the ICA method provides systematic procedures to extract the system ranges based on existing product cases. Following the design reuse rationale, historical data can be used to construct the system ranges using statistical methods. This can alleviate the need to analyze the design problem and extract system ranges from scratch.

It should be noted that because the two methods are defined at different levels of abstraction, the major difference concerning the establishment of system range lies in the different problem formulation. Hence, it is not appropriate to compare the two methods in terms of accuracy and effectiveness for deriving the system ranges based on a common case study. However, the research does provide a case study to compare the effectiveness of product performance evaluation using the ICA method and the experience-based method (Section 7.4). This can partially validate the effectiveness of the ICA method.



### 5.3 Precautions And Limitations

As can be seen from the ICA process, the performance of a new product is determined from the performance of existing similar products based on the principle of design-by-analogy. The relationships between the performance and the design parameters are established using the correlation matrices. In comparison to equation-based or meta-model-based approaches, the correlation-matrix-based relationship is not straightforward and conceals some design details. The research advocates this method for the following reasons.

- (1) At the early design stage, it is very difficult to build the equations or meta-models that are required to carry out the evaluations, mainly due to the paucity of the information. Even if such relations can be established, the uncertainties and variations in design make their application questionable (Ulrich and Eppinger, 2004). Therefore, it is advisable to make the estimations according to the availability of information.
  
- (2) A major concern of performance evaluation is the ultimate performance outcome instead of the details of component attributes. In comparison with the detailed *content* of the design parameters, the *consequence* is of greater importance (Jiao and Tseng, 2004). This is actually the essence of design reuse and design-by-analogy, in which the need for detailed information of the component attributes and their relationships can be avoided.

In Step 5 of the ICA method, one situation has been overlooked, i.e., the design synthesis may generate component combinations which capability indices have not been defined in Step 3. This is because the capability indices have been established according to existing product cases, and all the possible component combinations may not have been used in these product cases. In such a situation, the newly generated component combination (denoted as  $A$ ) is compared with all the component combinations, which capability indices have already been defined. The combination that is most similar to  $A$  is selected. Similarity is measured based on the major attributes of a component, i.e., the parameters that determine the characteristics of a component. The capability index of the selected component combination is used as the capability index of  $A$ .

In this thesis, the summation of the individual information content with respect to each and every KC is taken as the information content of the product. Strictly speaking, this is only applicable when the KCs are independent of each other. When this condition is not met, errors may occur, which may affect the validity of the subsequent design evaluation. According to the rationale of this research, the KCs denote the most important factors that signify the performance of a product family. In such a sense, each KC represents a unique feature of the product, such that the interdependency between KCs is insignificant. However, strict independence between KCs may be difficult to be achieved, especially for variant design (Jiao and Tseng, 1998). Therefore,

the summation of individual information content is only an approximation of the information content of the product.

#### **5.4 Summary**

This chapter proposes the ICA method for product performance evaluation. This method comprises of systematic procedures to compute the information content of a product, which is used as a uniform metric of product performance. It emphasizes the establishment of the system ranges in terms of the component capability indices based on existing product cases. Thus, it is in line with the rationale of design reuse.

The product data has been collected and analyzed, culminating in the establishment of a product platform. The knowledge embedded in the product platform is expected to be utilized in the design of product family configurations, which is formulated as an automated design synthesis process to be discussed in Chapter 6.

# Chapter 6    MULTIPLE OBJECTIVE OPTIMIZATION FOR DESIGN SYNTHESIS

This chapter proposes a multiple objective optimization method to carry out design synthesis and evaluation. Product cost and performance, which are the major concerns of product family design, are used as the optimization objectives. The optimization method is capable of automated generation and evaluation of product configurations, leading to the optimal solutions of a product family. A weighted summation method is proposed for post-optimal solution selection.

## 6.1 Problem Formulation

In this research, design synthesis is formulated as a configuration design problem. A general configuration design task is characterized by: (1) the generation of solutions (2) to satisfy a set of design requirements, based on (3) “a fixed, predefined set of components, where a component is described by a set of properties and ports for connecting it to other components” (Mittal and Frayman, 1989). For the product family design problem, the design requirements are defined according to the KCs, to address the target market segmentations. The physical components in the component catalog are identified in Stages I and II of the PFDR methodology. Next, the configurations of a set of related products are generated, and evaluated with respect to cost and

performance, leading to optimal configurations of the product family. Through these processes, product ‘similarity’ and ‘variety’ can be addressed simultaneously. They are managed as an optimal trade-off between product commonality and performance. Commonality bespeaks the similarity among products, while the diverse performance shows the many possible product varieties. It should be noted that the varieties of products is defined by the designer, based on his/her knowledge of the market segmentations. However, such a decision is beyond the scope of the PFDR methodology. For a more comprehensive study of how to decide an optimal profile of product variants, one can refer to Chen *et al.* (1996), Simpson *et al.* (1997), and Grante and Andersson (2003).

Based on previous discussions, a component catalog is denoted as:  $\mathbf{m} = [s_1, s_2, \dots, s_m]^T$ , where  $s_i$  is the  $i^{\text{th}}$  component slot that corresponds to a function  $f_i$ , and  $s_i = [m_i^1, m_i^2, \dots, m_i^{n_i}]^T$ , where  $n_i$  is the number of components in the component slot. Using a more general form, the components can be collectively represented as:  $\mathbf{m} = [m_1, m_2, \dots, m_n]^T$ , where  $n$  is the total number of physical components.

Thus, the components  $m_1, m_2, \dots, m_n$  are considered as the design parameters, which constitute a parameter space  $S$ , and  $S \in R^n$ . Due to the constraints imposed by the compatibility among components, an arbitrary configuration may not be feasible. Thus, the parameter space can be divided into two regions, namely, feasible and infeasible regions. An optimal solution must also be feasible.

The desired performance can be specified according to the KCs, which are denoted as  $\mathbf{k} = [k_1, k_2, \dots, k_p]^T$ . For each possible product configuration, the performance with respect to the KCs can be computed. The results are then aggregated into the information content ( $I$ ), which can be computed using the ICA method. At the same time, the cost ( $C$ ) of the product family can be computed using the predefined cost model. In an effort to achieve optimal trade-offs between performance and cost,  $I$  and  $C$  are used as two objective functions to be optimized. They can be represented as  $T(\mathbf{m}) = [I(\mathbf{m}), C(\mathbf{m})]$ , where  $T$  defines a 2D attribute space, i.e.,  $T \in R^2$ . Thus, the product family design problem is formulated as a multiple objective optimization problem (MOOP) (Figure 6.1).

$$\text{Min. } T(\mathbf{m}) = [I(\mathbf{m}), C(\mathbf{m})]$$

$$\text{s.t. } \mathbf{m} \in |S|$$

where:

$I(\mathbf{m})$  is the average information content of the product family.

$C(\mathbf{m})$  is the cost of product family.

$\mathbf{m}$  is a vector containing the physical components. Its elements are considered as the design parameters, which constitute the parameter space  $S$ .

$|S|$  is the feasible design space,  $|S| \subset S$ . Feasibility is defined with regard to the compatibility among the physical components.

Figure 6.1 Problem formulation of design synthesis and evaluation

Based on problem formulation, one objective of product family design is to reduce the

total production cost. An accurate and logical estimation of the product family cost directly influences the effectiveness of the method. The cost model is introduced first. Next, the algorithm to solve the optimization problem will be presented in Section 6.3.

## **6.2 Establishment Of Product Family Cost Model**

To develop an empirical cost model, there are two essential and interrelated factors, namely, the cost structure and the cost modeling method. The cost structure refers to the constitutive elements of production cost. The cost modeling method is the method to aggregate these cost elements into a mathematical form. The cost model for the PFDR methodology is designed. It should be noted that the cost model developed in this section involves techniques to extract cost elements from existing products, which corresponds to the cost modeling operator,  $Op_c$ .

### **6.2.1 Cost structure and cost model**

Basically, the cost elements can be considered from two perspectives (Michaels and Wood, 1989): (1) the top-down perspective, in which cost estimation is based on various high-level product utilities, such as product function and performance characteristics (e.g., weight, speed, reliability, etc.); and (2) the bottom-up perspective, in which cost estimation is based on the basic manufacturing elements, such as material, labor, and energy consumption. These elements can be further divided into different forms, such as direct or indirect costs, and variable or fixed costs (Hundal, 1997). Measurements of these elements can be time, operation, weight, etc. Several

representative cost modeling methods are summarized in Table 6.1.

Table 6.1 A few representative cost models

Method	Characteristics
Function costing (Hundal, 1997)	Cost is attributed to the product functions. This method is suitable for cost estimation at the conceptual stage.
Parametric costing (Michaels and Wood, 1989)	Also known as cost estimating relationships (CERs). Statistical techniques such as regression or correlation analysis are carried out with respect to one or more product attributes, such as weight, power, etc.
Productive hour costing (Ostwald, and McLaren, 2004)	Indirect costs are first converted to equivalent production hours. Then, the productive hour cost rate is determined by allocating indirect costs (overhead) to each hour of manufacturing process. The overall indirect cost is the multiplication of the above items.
Activity-based costing (ABC) (Cooper and Kaplan, 1991)	This method identifies the activities that drive costs by consuming resources. Typical activities include: number of units produced, cost of materials used, number of different materials used, labor hours, hours of equipment time, number of orders received, etc.
Magnitude-based costing (Hundal, 1997)	Costs are assigned to product components. These components are divided into three grades according to specific properties (e.g., weight, cost). A higher grade has a larger impact on the overall cost. Thus, the importance of the components in costing estimation is determined.

In product family design, cost estimation is usually presented as a combination of these modeling techniques. Gonzalez-Zugasti *et al.* (2001) proposed a cost model that accounts for the investment on platforms and product variants. Uncertainties during the development of the product family are also considered. Chidambaram and Agogino (1999) proposed a model that combines the generational cost, which is based on the fundamental elements and features of the product, and function-based cost. Regression-based and similarity-based methods are used to compute the cost. Fujita *et al.* (1999) developed a cost model by considering the fixed and variable costs.



Learning effect is considered when computing the fabrication cost and the assembly cost. However, these methods require a lot of downstream manufacturing information which may not be available at the product planning stage. Thus, a number of coefficients and parameters have been assigned subjectively. Finally, Park and Simpson (2005) proposed a comprehensive cost model based on activity-based costing. However, this model is restrictive in that production activities and resource data are usually not known to designers at the early design stage.

### 6.2.2 An empirical cost model for product family design

In this research, a cost model that considers three elements, namely, fixed cost, development cost and component cost was developed. These elements are defined in different levels of the product family development (Figure 6.2). Accordingly, the cost model in mathematical form is presented in Equations (6.1) and (6.2).

$$C = C_F + \sum_{i=1}^N C_P^i \quad (6.1)$$

$$C_P^i = C_d^i + \sum_{j=1}^{M^i} C_c^j \quad (6.2)$$

where:

$C_F$  is the fixed cost at the corporation or department level.

$C_P^i$  is the cost of product  $i$ .

$N$  is the total number of products in the product family.

$C_d^i$  is the development cost of product  $i$ .

$C_c^j$  is the cost of component  $j$ .

$M^i$  is the number of components implemented in product  $i$ .

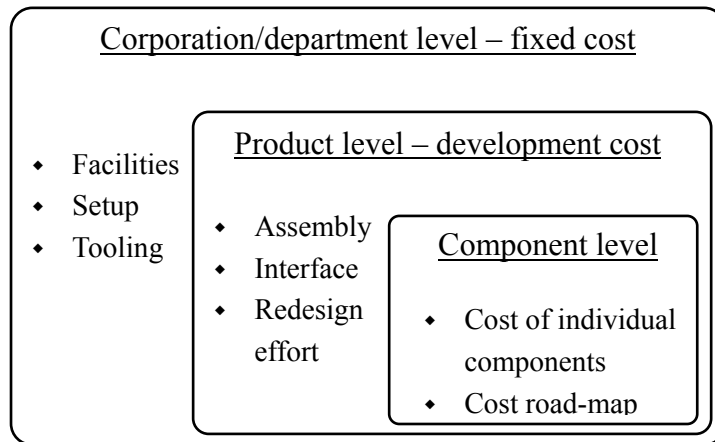


Figure 6.2 Cost model of a product family

### 6.2.2.1 Component cost

The component cost is the most important part of the cost model. It accounts for the majority of the product cost. According to the design reuse methodology, the physical components are the elementary building blocks of the product family. Thus, the component cost can be used as a basic costing element. In this way, the lower level product cost information, such as the primitive manufacturing elements (e.g., material, machining, labor, etc.), is not required. In addition, if the components are outsourced to third party manufacturers, the cost can be estimated accurately. By analyzing the historical data, cost road-maps can be built for the components to indicate the trend of the cost evolution. The production volume is an important factor that influences the unit cost of a component. Moreover, cost usually decreases with time as new manufacturing technologies are constantly being introduced. As an example, the cost road-maps of two cellular phone batteries are illustrated in Figure 6.3. Each curve

represents a road-map of the unit cost of a component at a particular production volume per year (the number shown on top of the curve).

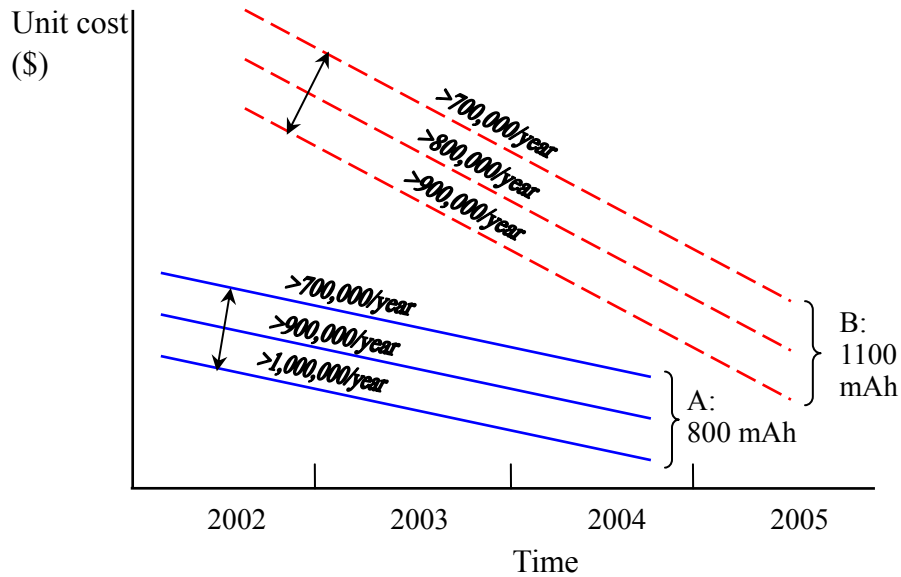


Figure 6.3 Cost road-maps of cellular phone batteries

### 6.2.2.2 Product development cost

The development cost is applied at the product level. Apart from the cost of individual components, cost is incurred to assemble the components, or to analyze and test the performance of the products. Factors to be considered include the characteristics of the interfaces, the complexity of the assembly, and the fraction of a design that is new. Different criteria can be applied to estimate the development cost, such as the development time (Grante and Andersson, 2003), number of primitives (Fujita *et al.*, 1999), product weight (Hundal, 1997; Fujita *et al.*, 1998), etc.

This research adopts a model that accounts for the complexity caused by the number of

components ( $N_c$ ), the number of types of components ( $N_t$ ), and the number of interfaces ( $N_i$ ). First, the complexity of a product is estimated based on these factors. Next, this complexity is used to approximate the development cost. The development cost of product  $i$  is calculated as (Meyer and Lehnerd, 1997):

$$C_d^i = \alpha^i \sqrt[3]{N_c N_t N_i} \quad (6.3)$$

where:

$\alpha^i$  is the cost coefficient of product complexity.

$\sqrt[3]{N_c N_t N_i}$  is used to measure the product complexity.

### 6.2.2.3 Fixed cost

The fixed cost is estimated at the corporation or department level. It includes the costs required to maintain the regular operation of a manufacturing unit, such as the cost incurred for the maintenance of the manufacturing facilities, the setup and tooling costs, etc. This part of the cost is not sensitive to the changes of the product configurations, and remains constant over a period of time. The fixed cost is estimated based on the historical manufacturing data. It should be noted that, in this research, the fixed cost refers to the fixed cost of a product family. If several product families are being designed simultaneously, it is necessary to apportion the fixed cost among them. However, the development of a very comprehensive cost model for multiple product families is not the focus of this research and hence is not discussed here.

This section presents a cost model for product family design. Analysis of existing

products and manufacturing systems is emphasized in this cost model, which makes it suitable to the design reuse methodology. The cost elements can be refined with respect to a specific product family to be designed. The overall cost of a product family is estimated based on the model once the configurations of the products are generated.

### **6.3 Multiple Objective Optimization**

Based on the problem formulation in Section 6.1, the design problem is reduced to the selection and combination of the physical components from the component catalog to minimize the design objectives. In a fully-developed design database, sufficient number of components will be retrieved as building blocks. Thus, there will be many possible combinations. Moreover, for a typical engineering design problem, little is known about the shape and modality of the attribute space *a priori*. For example, the objective functions can be linear or nonlinear; the attribute space can be convex or non-convex, and discrete or continuous. This is especially true for a generic method applicable to the design of various types of products. Although the PFDR method has restricted the objective functions to information content (I) and cost (C), which characteristics are partially known, a robust and efficient search and optimization algorithm is still valuable to make the design synthesis efficient and flexible.

### 6.3.1 Introduction of multiple objective optimization problem

A number of multiple objective optimization algorithms have been reported. Some of these algorithms have converted a MOOP into a single objective optimization problem using a few user-defined procedures, such as the weighted sum method, the  $\epsilon$ -constraint method, the weighted metric method, etc. (Deb, 2001). Deb (2001) pointed out that a major limitation of these methods is that they involve a number of user-defined parameters which are difficult to set in an arbitrary problem.

GA has been advocated by many researchers due to its power in solving complex combinatorial problems, especially for the design synthesis problem. In comparison with other methods, such as linear programming and nonlinear programming, GA is derivative-free, i.e., it does not require the objective functions to be derivable with respect to the design parameters (Simpson, 2004). Therefore, it can deal with the discrete design space with ease. Moreover, a group of solutions (population) is maintained during the search process. Hence, it is possible to obtain multiple solutions. This is especially useful for designers who want to get a few candidate solutions, with each excelling in certain aspects. Multi-objective optimization based on GA has been extensively studied and reported to address the above deficiency (Deb, 2001). These algorithms are usually divided into the non-Pareto and Pareto-based approaches (Fonseca and Fleming, 1998). Andersson and Wallace (2002) proposed a Pareto-based approach, namely, the multi-objective struggle genetic algorithm (MOSGA), and compared it with a few typical multi-objective GA, such as Vector Evaluating GA

(VEGA), non-dominated sorting GA (NSGA), multi-objective GA (MOGA), etc. It is claimed that the MOSGA method can handle the multi-modal attribute spaces with improved robustness. Moreover, it requires less tuning of the GA parameters in comparison with the other methods. These are desirable for the design synthesis problem. Therefore, this research adopted the MOSGA method.

In a MOOP, the optimization results usually constitute a Pareto-optimal solution set instead of just one optimal solution. A Pareto-optimal solution is also called a non-dominated solution. A solution is said to dominate another if it is superior in most, if not all objectives. Consider a minimization problem  $T(\mathbf{m})$  with  $r$  objectives, and two solution vectors  $\mathbf{m}_1$  and  $\mathbf{m}_2$ ,  $\mathbf{m}_1$  is said to dominate  $\mathbf{m}_2$ , if

$$\forall i \in \{1, 2, \dots, r\} : T_i(\mathbf{m}_1) \leq T_i(\mathbf{m}_2) \quad \text{and} \quad \exists j \in \{1, 2, \dots, r\} : T_j(\mathbf{m}_1) < T_j(\mathbf{m}_2)$$

In the product family design problem, two objectives are involved, i.e.,  $r=2$ , and  $T_1(\mathbf{m}) = I(\mathbf{m})$ ,  $T_2(\mathbf{m}) = C(\mathbf{m})$ .

In a Pareto-based optimization GA, the fitness of an individual solution is represented as its rank. Each individual is assigned a rank, which is the number of population members that dominate over it plus one (Fonseca and Fleming, 1998). Therefore, a lower ranking indicates a better fitness, and the solutions with the rank '1' are the non-dominated solutions.

### 6.3.2 Multi-objective struggle genetic algorithm

Based on the above discussion, the MOSGA algorithm for the design synthesis problem is outlined next (Figure 6.4).

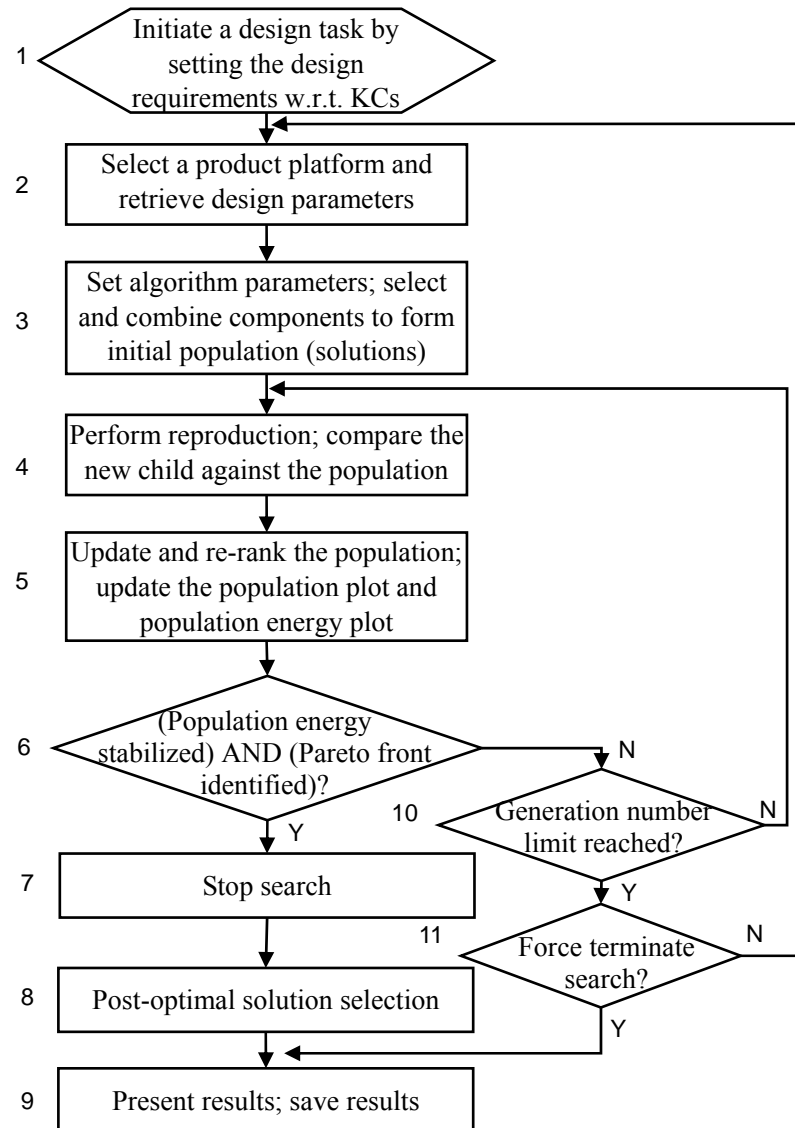


Figure 6.4 Flowchart of MOSGA for the design synthesis problem

**Step 1.** A design task is initiated by selecting a set of KCs and setting the design requirements of a product family with respect to these KCs. The design ranges are thus defined in this step.



- Step 2.** A product platform is chosen and the physical components are retrieved from the component catalog as the design parameters.
- Step 3.** The parameters of the algorithm are set up. In particular, the population size and the number of generations are the most important controlling parameters. The initial population is generated according to the population size.
- Step 4.** Two individuals are randomly selected from the population, and crossover and mutation are performed to generate the children. In this process, the stochastic universal selection, the single-point crossover, and the bit-wise mutation, which are common GA operators, are used to generate the offsprings. Next, the two objective functions of each child are calculated. A child is compared with every individual in the population with respect to the objective functions, through which the rank of the child is obtained.
- Step 5.** The algorithm searches for an individual that is most similar to this child and replaces it with the child if the child has a lower rank, or if the child dominates over it. The rank of the population is updated if the child has been inserted. The population energy and the population plot are updated accordingly.
- Step 6.** The Pareto-optimal front is identified in this step. It should be noted that the purpose of this method is to find adequate design variants, instead of identifying each and every global optimum. Therefore, once the design space has been explored sufficiently such that the population converges to a specific curve/surface, presumably the Pareto front, the search is stopped.

The mechanism to ensure a sufficient exploration of the design space is discussed in Section 6.3.3.3.

**Steps 7-11.** In the subsequent steps, the algorithm controls the search by checking whether the number of generations reaches a predefined limit, or whether the user force stops the search. The Pareto-optimal set, if it is obtained, undergoes the post-optimal selection process to arrive at the final candidate solutions. Design synthesis results are saved into the database if desired.

### 6.3.3 Important issues in the optimization algorithm

#### 6.3.3.1 The structure of a chromosome

In this research, a chromosome (i.e., a solution or a product family configuration) is a combination of the physical components. The length of a chromosome is the number of component slots ( $m$ ) multiplied by the number of product members in the product family ( $N$ ). Each bit on the chromosome string is an integer which is the index of a component in its component slot ( $s_i$ ). The structure of a chromosome is illustrated using a 2D array as is shown in Table 6.2. The first column lists the component slots. By linking the numbers in all columns in sequence, a string of bits is formed to indicate the selection of the components. For example, the chromosome of the electric kettle product family is: 231553111221613137142 (the strings of  $P_1$ ,  $P_2$  and  $P_3$ ).

Table 6.2 Chromosome structure of the electric kettle product family ( $N=3$ )

Component slot		$P_1$	$P_2$	$P_3$
$s_1$	Insulator	2	1	3
$s_2$	Contact disk / radiator	3	1	1
$s_3$	Outlet device	1	2	3
$s_4$	Heating disk / coil	5	2	7
$s_5$	LED / LCD screen	5	1	1
$s_6$	Container	3	6	4
$s_7$	Thermometer and switch	1	1	2

### 6.3.3.2 Measurement of similarity between individuals

In Step 4 of the MOSGA algorithm, a measurement of the similarity between the individual chromosomes is required. The similarity between two individuals is measured using the combined distances in the attribute space ( $T \in R^2$ ) and the parameter space ( $S \in R^n$ ). The distances are calculated as follows.

Two individuals,  $\mathbf{a}$  and  $\mathbf{b}$ , are denoted as:

$$\mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mN} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mN} \end{bmatrix},$$

where  $a_{ji} = x$ , if component  $m_j^x$  is selected to form the chromosome with respect to the corresponding component slot  $s_j$ . A similar definition is adopted for  $\mathbf{b}$ .

The distance between  $\mathbf{a}$  and  $\mathbf{b}$  in the parameter space is computed as:

$$D_{IST}^S = 1 - \frac{\|\mathbf{a}, \mathbf{b}\|}{m \times N} \quad (6.4)$$

where:

$m$  is the number of component slots.

$N$  is the number of products in the product family.

$\|\mathbf{a}, \mathbf{b}\|$  is the total number of elements that satisfy  $a_{ji} = b_{ji}$ , ( $i = 1, \dots, N; j = 1, \dots, m$ ).

Thus,  $0 \leq \|\mathbf{a}, \mathbf{b}\| \leq m \times N$ . It follows that  $0 \leq D_{IST}^S \leq 1$ , where ‘0’ indicates that two individuals use the same set of components and ‘1’ indicates that they are completely different.

The distance between  $\mathbf{a}$  and  $\mathbf{b}$  in the attribute space is computed as:

$$D_{IST}^T = \frac{\sqrt{\sum_{i=1}^2 (T_i(\mathbf{a}) - T_i(\mathbf{b}))^2}}{\|T\|} \quad (6.5)$$

The numerator is the Euclidean distance of two individuals in the attribute space and  $\|T\|$  is the diameter of the attribute space. Since the shape of the attribute space is not known *a priori*, the exact value of  $\|T\|$  is not available. Therefore, it is approximated as the maximum distance between two individuals in the current population. Based on this formulation,  $0 \leq D_{IST}^T \leq 1$ .

Finally, the combined distance is computed as

$$D_{IST} = \frac{1}{2} (D_{IST}^S + D_{IST}^T) \quad (6.6)$$

### 6.3.3.3 Ensuring sufficient exploration of the design space

Step 6 requires that the design space be explored sufficiently to obtain the optimal or near optimal solution. Two simple criteria are presented next. A sufficient exploration of the design space is confirmed when both criteria are satisfied.

(1) The average population energy is stabilized at a reasonably low level. The average population energy refers to the average values of the objective functions according to the current population. The energy is calculated for each generation. It will decrease gradually, and converge to a relatively stable state after a certain number of generations. It is observed that few new Pareto optima are discovered after the search reaches the stable state, and hence this is an indication that the design space has been effectively explored. A typical population energy curve is shown in Figure 6.5.

(2) The Pareto-optimal front can be identified in the population plot. A population plot illustrates the individuals as points in the attribute space, with their objective functions as the coordinates. During the GA-based search, new individuals will be added into the plot. It is observed that at the early stage of the search, a number of new individuals are produced. As the search proceeds, fewer new individuals are produced, and the non-dominated optimal solutions gradually form a specific curve/surface. When such a curve/surface becomes apparent, the search is stopped manually by the designer. A typical population plot is shown in Figure 6.6. In this

figure, the diamonds represent the initial population and the crosses represent the individuals generated during the search. The search converges to a Pareto-optimal set, indicated by the triangles.

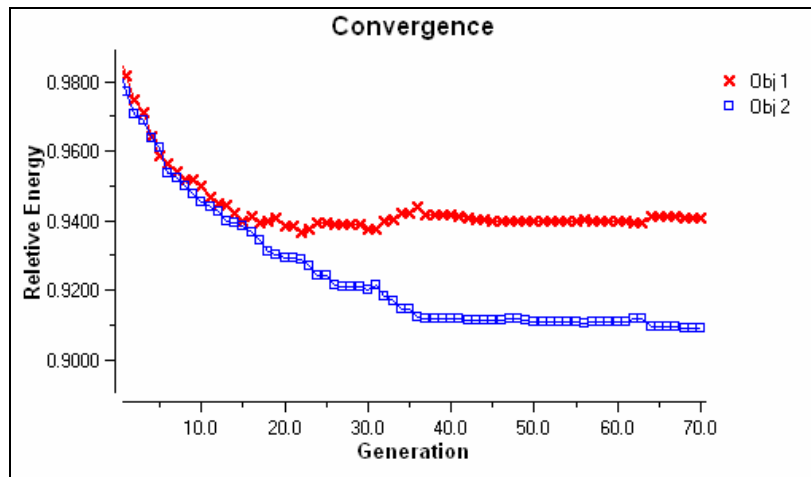


Figure 6.5 Visualization of population energy convergence

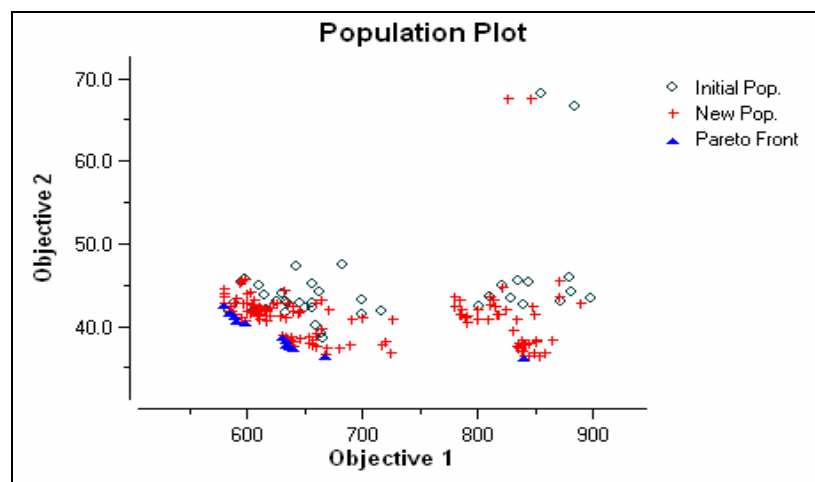


Figure 6.6 Visualization of population energy and population plot

A design synthesis system has been developed. It is capable of visualizing the population energy and population plot. The user can observe the search process, stop

the search, or fine-tune the parameters with a new iteration. For the parameters fine-tuning, two parameters are essential, namely, the population size and the number of generations. A user can carry out a few tentative design synthesis sessions to determine these parameters. Generally, the design space can be sufficiently explored with a large population size and a large number of generations. However, this will significantly increase the computational load. A practical method is to start from a relatively small population size and a small number of generations, and increase them gradually. The parameters are finalized when the increase of the two parameters does not result in an obvious improvement of the Pareto front.

#### 6.4 Post-Optimal Solution Selection

The MOSGA method will result in a Pareto-optimal set. Usually, a designer needs to select one or more candidate solutions to be further developed in the subsequent design processes. To do so, some meta-knowledge is needed, e.g., a designer may need to show his/her preference of the design objectives, namely, performance or cost, through assigning different weights to them. This leads to configurations with different priorities to performance and cost. In this research, a relative weight is assigned to each objective and a weighted fitness  $T_w(\mathbf{m})$  is obtained for every Pareto-optimal solution.

$$T_w(\mathbf{m}) = w_1 \frac{I(\mathbf{m}) - I^{\min}}{I^{\max} - I^{\min}} + w_2 \frac{C(\mathbf{m}) - C^{\min}}{C^{\max} - C^{\min}} \quad (6.7)$$

where:

$w_1$  is the weight assigned to the information content, and  $w_2$  is the weight assigned to cost, and  $w_1 + w_2 = 1$ .

$I^{\max}$  and  $I^{\min}$  are the maximum and minimum information contents.

$C^{\max}$  and  $C^{\min}$  are the maximum and minimum costs.

Figure 6.7 illustrates a Pareto-front, where each black square denotes a solution of a product family with information content and cost as the coordinates.  $(I^{\min}, C^{\min})$  constitutes the coordinates of the individual best objective function values. Since this solution is usually non-existent or infeasible, it is considered an utopian ideal solution. Thus,  $T_w(m)$  is a normalized, weighted distance that a solution is to the ideal solution. The final solution is chosen as the one(s) with the smallest weighted fitness. If the weighted fitness values of several solutions are very close, additional meta-knowledge is required to assess the merits of the candidate solutions.

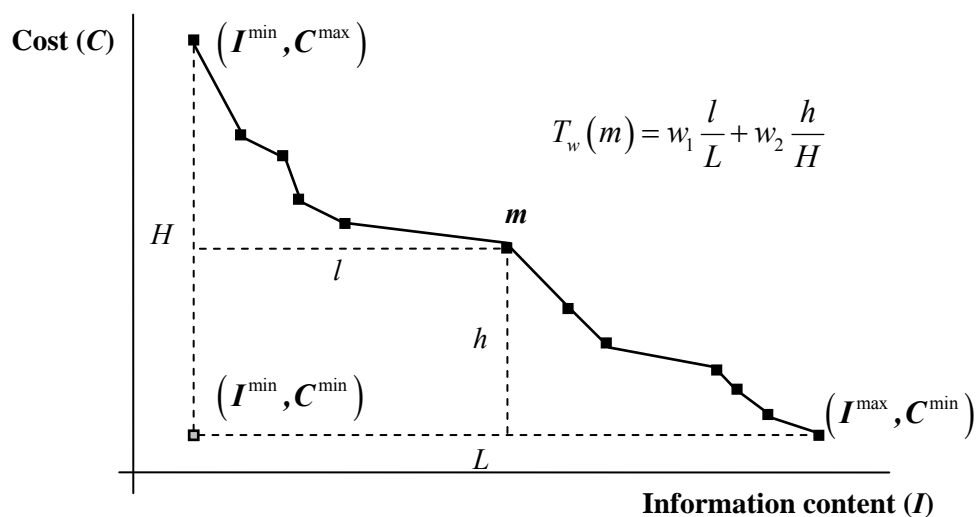


Figure 6.7 Pareto-front and post-optimal solution selection



## **6.5 Summary**

This chapter presents the automated design synthesis and evaluation method for product family design. First, the design synthesis problem is formulated as a multiple objective optimization problem. Next, an empirical cost model is established to compute the cost of the product family. The MOOP is solved using a GA-based search algorithm. The design synthesis and evaluation process is built upon the results and techniques of previous stages, such as the product architecture, and solution evaluation based on the ICA method. Solutions are finalized through the post-optimal solution selection process according to the designer's preferences. Product family design is accomplished when the final solutions are generated and evaluated.

# Chapter 7    SYSTEM IMPLEMENTATION AND CASE STUDIES

Chapters 3 ~ 6 establish the theoretical foundation of the PFDR methodology. In this chapter, the implementation issues are discussed. Three case studies illustrating different scenarios are presented. The effectiveness of the PFDR method is discussed accordingly.

## **7.1 A Prototype Product Family Design Reuse System**

A prototype system was developed to implement the PFDR methodology. Figure 7.1 shows the architecture of the system. This system emphasizes the processing and transformation of product data across different stages of the PFDR process model. The central knowledge base acts as a container of both raw product data and the refined knowledge obtained through knowledge extraction. It is the source of information for design synthesis. The processing power of the system resides in three core processing engines (the shaded rectangles), namely, the information modeling engine, the knowledge extraction engine and the solution generation engine. These processing engines are supported by various techniques and algorithms. The processing engines accept requests from the designers, such as data entry, knowledge extraction, and design synthesis, trigger the corresponding computational tools and communicate with

the central knowledge base to retrieve data.

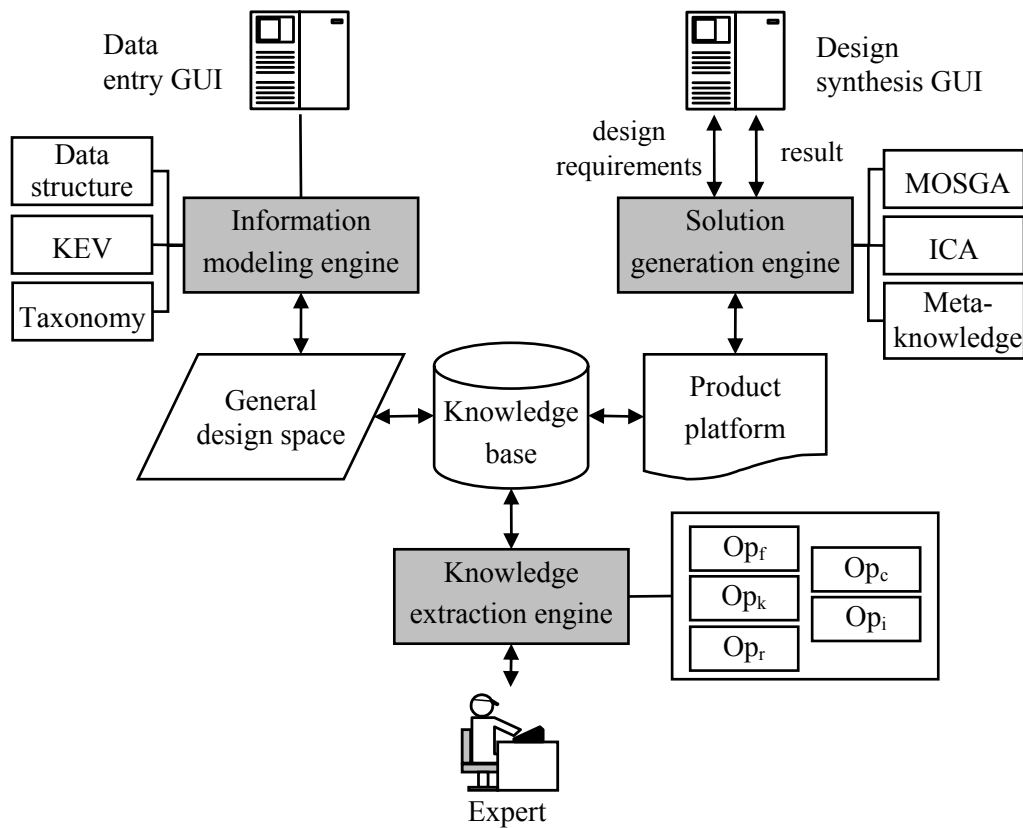


Figure 7.1 Architecture of the PFDR prototype system

The data entry GUI is designed to allow for easy entry of product information (Figure 7.2). The multiple facets of product information can be modeled using predefined structures. For example, the function structure of a product can be constructed using the GUI shown in Figure 7.3. With the support of taxonomy, function decomposition can be carried out using simple ‘click-and-assemble’ operations. This can significantly alleviate the effort to model product functions.

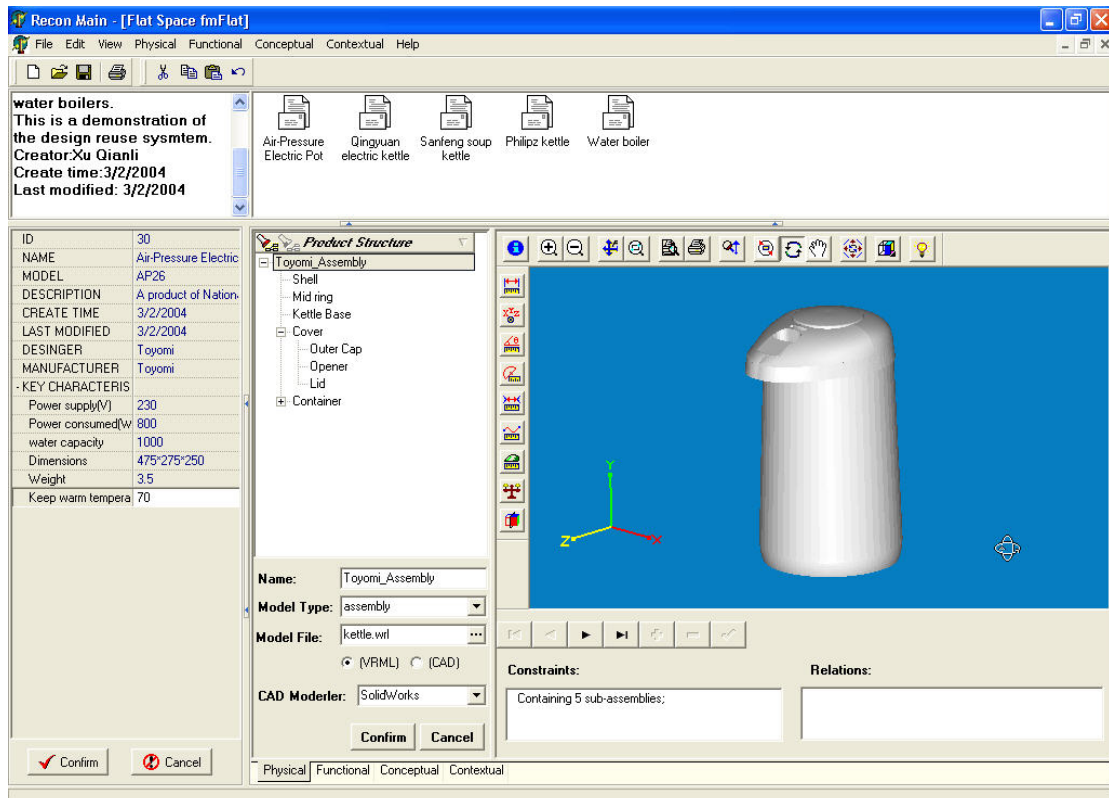


Figure 7.2 User interface for product information modeling

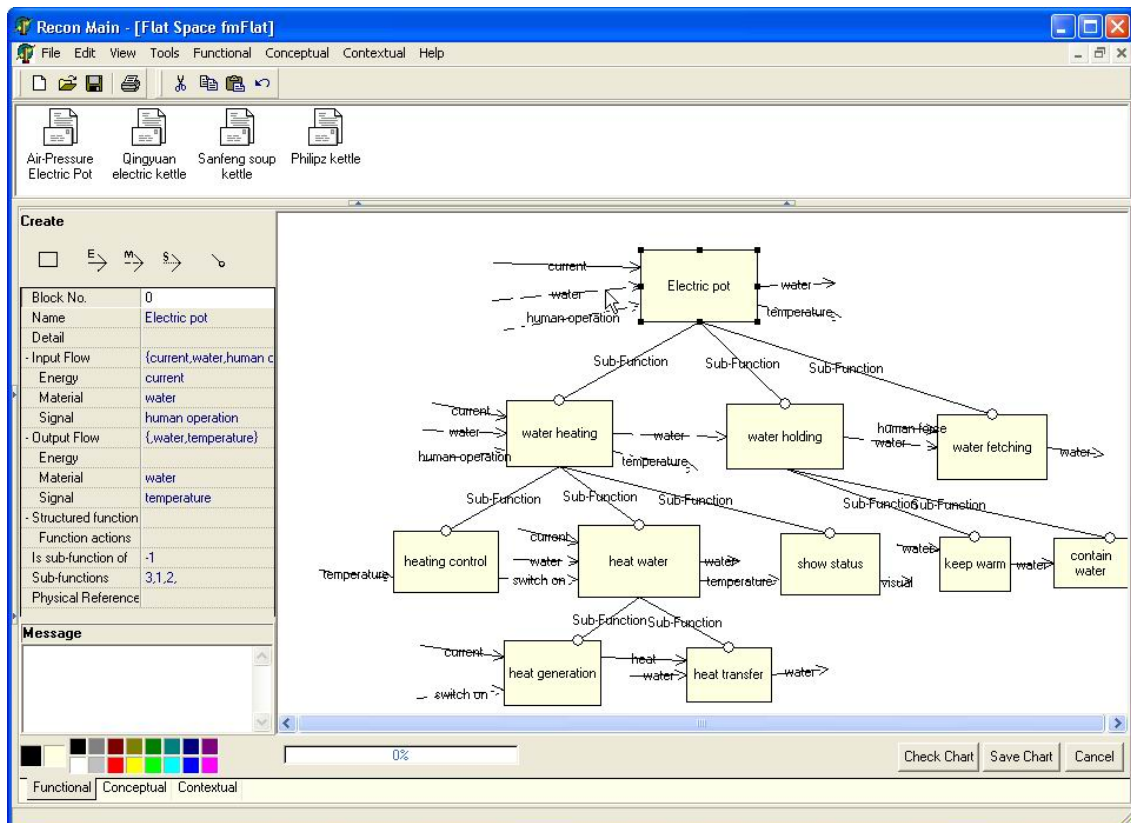


Figure 7.3 User interface for product function decomposition

The raw product data are stored in the knowledge base. When sufficient product cases have been collected, the knowledge extraction engine is triggered to allow an expert to reconfigure the raw product data using the knowledge extraction operators  $\{Op_x\}$ . These operators are supported by various techniques, either computationally or manually.

At the design synthesis and evaluation stage, a designer can input the design requirements to define a product family. The number of products in the product family, as well as the design requirements, are input manually. That means the variety of the products in a product family is defined by the designer, based on his/her knowledge of the market segmentations. Next, the PFDR system will generate the configurations of the product variants using the predefined components, and achieve an optimal trade-off between product performance and cost. Automated design synthesis is carried out using the design synthesis engine, which is driven by the MOSGA and the ICA method. MOSGA can cater to any number of products that a designer inputs. However, the algorithm itself does not decide the optimal number of products in the product family. The GUI for design synthesis is shown in Figure 7.4.

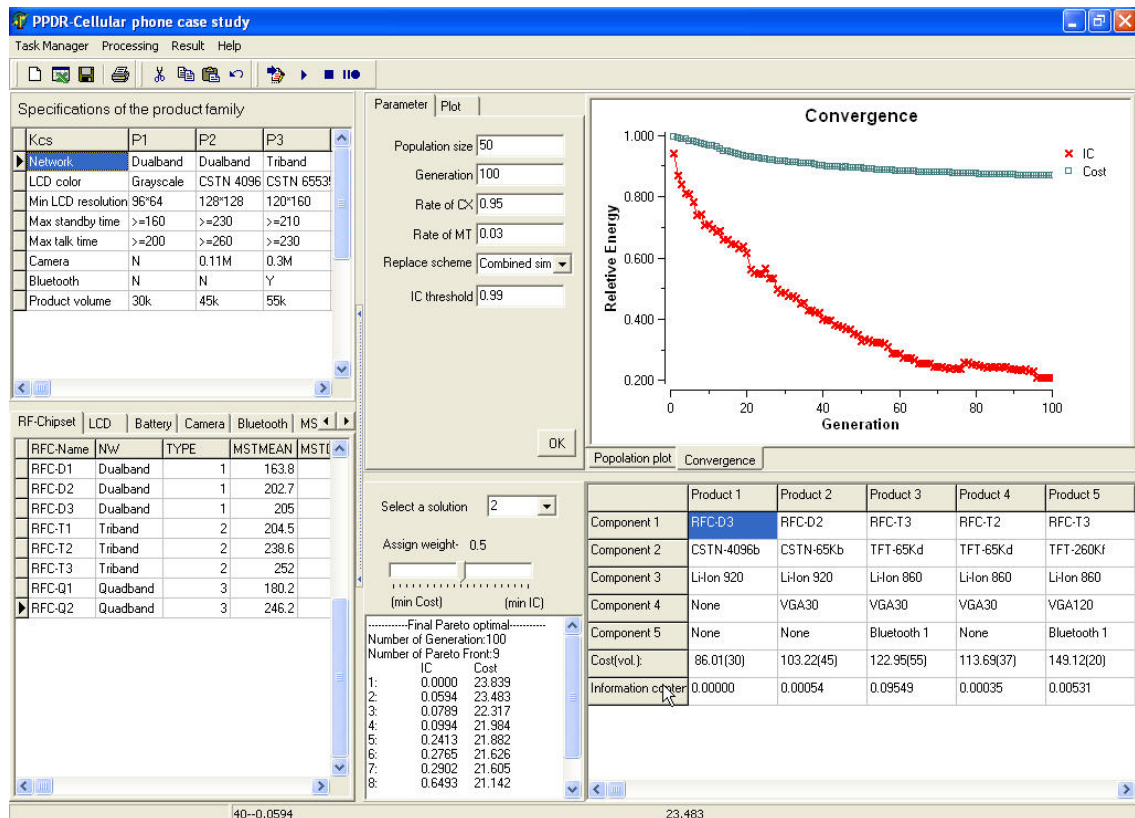


Figure 7.4 User interface for design synthesis

## 7.2 Case Study I: Cellular Phone Product Family Design

This case study focuses on the design of a family of cellular phone products. The purpose of this case study is to show the effectiveness of the PFDR methodology in automated design synthesis and evaluation. In particular, it shows how trade-offs between cost and product performance are managed using this method. The cellular phone is a typical consumer product. It has a modular structure and many reusable components. Product performance can be customized using components with different properties. A product family is designed as an optimal trade-off between the performance and cost of the product.

### 7.2.1 Settings

96 different models of cellular phones produced by the same manufacturer were collected. The products were analyzed and modeled according to product functions, KCs, and physical components. The function structures were similar among different product models, and many functions have been implemented using standard physical components. The properties of the components, e.g., cost and major technical attributes, were collected and documented.

The KCs that address the major customer requirements were extracted (Table 7.1). An FPA was established based on the individual function structures. The relationships between the KCs and the product functions were established using a correlation matrix,  $TR_2$ , in Table 7.2. This research used a subset of KCs and functions in the case study for brevity. The component slots that correspond to product functions are shown in Table 7.3. The physical components belonging to actual product cases were assigned to the component slots.

Table 7.1 KCs of the cellular phones

KCs	Abbr.	Type	Unit	Default value
Network	NW	categorical	–	Dualband, Triband, Quadband
Display color	DC	categorical	–	grayscale, CSTN4096, TFT65k ...
Display resolution	DS	categorical	pixel	96×64,128×128,128×160,176×220...
Max talk time	MTT	real	hour	–
Max standby time	MST	real	minute	–
Built-in camera	CM	ordinal	mega pixels	0, 0.11, 0.3, 1.2
Bluetooth connection	BC	Boolean	–	True, False

Table 7.2 Correlation between KCs and functions ( $TR_2$ )

KCs		$f_1$	$f_2$	$f_3$	$f_4$	$f_5$
$k_1$	NW	1	0	0	0	0
$k_2$	DC	0	1	0	0	0
$k_3$	DS	0	1	0	0	0
$k_4$	MTT	1	0	1	0	0
$k_5$	MST	1	1	1	0	0
$k_6$	CM	0	0	0	1	0
$k_7$	BC	0	0	0	0	1
$f_1$ : Signal processing		$f_2$ : Display		$f_3$ : Power supply		$f_4$ : Image capturing
$f_5$ : I/O connection						

Table 7.3 Function and component slot

Function		Component slot		Number of Components
$f_1$	Signal processing	$s_1$	RF chipset & base band	8
$f_2$	Display	$s_2$	LCD screen	16
$f_3$	Power supply	$s_3$	Battery	7
$f_4$	Image capturing	$s_4$	Camera	5
$f_5$	I/O connection	$s_5$	Bluetooth set	3

Based on previous discussions, the cost road-maps and capability indices of the components have to be established. Basically the cost road-maps are available from market analysis and historical data. Since such information is proprietary and highly confidential, the actual cost data was not used in this research. Instead, the cost road-map was simulated using a simplified model.

$$C_c^j = C_c^0 \left( \frac{V_j}{V_0} \right)^{R_j} \quad (7.1)$$

where:

$C_c^j$  is the cost of component  $j$ .

$C_c^0$  is the cost at a given production volume  $V_0$ .



$V_j$  is the volume of component  $j$  used in the product family.

$R_j$  is a regression coefficient, which simulates the learning effect of production.

$R_j = \log_2 s$ , where  $s$  is the slope of the learning curve. Typical in electronics industry,  $90\% \leq s \leq 95\%$ , and hence,  $-0.15 \leq R_j \leq -0.074$  (Hundal, 1997).

The cost at the department level was fixed. Moreover, the cost at the product level was computed according to Equation (6.3) based on the complexities of different products.

The component capability indices that correspond to the KCs:  $NW$ ,  $DC$ ,  $DS$ ,  $CM$ , and  $BC$ , were easily established because each KC is related to only one function. On the other hand, the  $MST$  and  $MTT$  are related to more than one function. In particular,  $MST$  is related to three functions:  $f_1$  (signal processing),  $f_2$  (display) and  $f_3$  (power supply), and  $MTT$  is related to two:  $f_1$  and  $f_3$ . The capability indices of component combinations were established using statistical methods. For example, with respect to  $k_4=MTT$ , the component combination (RFC-T1, Li-Ion 860) has been used in five host products  $\mathbf{h} = [h^1, h^2, h^3, h^4, h^5]$ . Ten samples were retrieved for each host product and the  $MTT$  values of the samples were available from historical data (Table 7.4). The capability index was defined as a normal distribution:

$$\{(\text{RFC-T1, Li-Ion 860}) | MTT : \mu = 226, \sigma = 25.6\}.$$

Similar procedures can be carried out for the other component combinations.

Table 7.4 Sampled *MTT* values of five products that host (RFC-T1, Li-Ion 860)

Sample	$h^1$	$h^2$	$h^3$	$h^4$	$h^5$
1	249	209	273	242	216
2	238	237	233	194	225
3	260	189	254	255	237
4	237	228	236	208	201
5	222	205	275	196	201
6	218	246	259	227	240
7	230	210	227	184	216
8	250	224	253	234	219
9	251	231	239	209	242
10	228	224	247	221	223

Six products are to be launched as a product family. The design requirements are listed in Table 7.5, with the last row showing the production volumes. The MOSGA method was used to generate the optimal product family. After a set of Pareto-optimal solutions were generated, different weights were assigned to the two objectives, namely, cost and information content. Information content is used to measure the performance of a product, as is discussed in Section 5.2. Thus, the product family could accommodate the designer's preference on reducing cost or enhancing performance. For example, if a larger weight is assigned to the information content, the selection of the optimal solution is more sensitive to variations in the information content. As a result, a solution with a smaller information content (which means better performance) is selected. In this research,  $w_1$  is the weight assigned to performance (information content), and  $w_2=1-w_1$  is the weight assigned to cost. Three strategies were adopted, namely, performance priority ( $w_1=0.8$ ,  $w_2=0.2$ ), cost priority ( $w_1=0.2$ ,  $w_2=0.8$ ), and equal priority ( $w_1=w_2=0.5$ ).

Table 7.5 Design requirements of the product family

KCs		$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$
$k_1$	NW	Dualband	Dualband	Triband	Triband	Triband	Quadband
$k_2$	DC	grayscale	CSTN4096	CSTN65535	TFT65535	TFT260K	TFT65535
$k_3$	DS	96×64	128×128	120×160	128×160	176×220	176×220
$k_4$	MTT	≥160	≥230	≥210	≥200	≥170	≥180
$k_5$	MST	≥200	≥260	≥230	≥250	≥200	≥190
$k_6$	CM	0	0.11M	0.3M	0.3M	1.2M	1.2M
$k_7$	BC	False	False	True	False	True	True
Production volume		30,000	45,000	55,000	37,000	20,000	15,000

### 7.2.2 Results

Table 7.6 ~ Table 7.8 show the product configurations generated using the PFDR method, according to three priority strategies. The last row in each table shows the objective functions, and the last column shows the number of different components that were used in a product family. Figure 7.5 illustrates the objective functions of the three solutions under the three priority strategies.

Table 7.6 Product configurations (performance priority)

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	#
$s_1$	RFC-D3	RFC-D2	RFC-T2	RFC-T2	RFC-T3	RFC-Q2	5
$s_2$	Grayscale	CSTN-4096b	CSTN-65Kc	TFT-65Kd	TFT-260Kf	TFT-65Ke	6
$s_3$	Li-Ion 780	Li-Ion 860	Li-Ion 700	Li-Ion 860	Li-Ion 600	Li-Ion 780	4
$s_4$	None	VGA11	VGA30	VGA30	VGA120	VGA120	3
$s_5$	None	None	Bluetooth 1	None	Bluetooth 1	Bluetooth 1	1
IC	0.000	0.033	0.001	0.000	0.116	0.008	–
Cost	71.94	89.90	116.32	116.03	150.06	152.80	–
<b>Average IC: 0.026    Total cost: 22.188(×10<sup>6</sup>\$)</b>							<b>19</b>

Table 7.7 Product configurations (equal priority)

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	#
$s_1$	RFC-D3	RFC-D2	RFC-T2	RFC-T2	RFC-T2	RFC-Q2	4
$s_2$	Grayscale	CSTN-4096b	CSTN-65Kc	TFT-65Ke	TFT-260Kf	TFT-65Ke	5
$s_3$	Li-Ion 860	Li-Ion 780	Li-Ion 780	Li-Ion 780	Li-Ion 780	Li-Ion 780	2
$s_4$	None	VGA30	VGA30	VGA30	VGA120	VGA120	2
$s_5$	None	None	Bluetooth 1	None	Bluetooth 1	Bluetooth 1	1
IC	0.000	0.0326	0.003	0.166	0.000	0.007	–
Cost	75.15	92.77	113.60	109.50	139.77	145.03	–
<b>Average IC: 0.035    Total cost: 21.7(<math>\times 10^6</math>S\$)</b>							<b>14</b>

Table 7.8 Product configurations (cost priority)

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	#
$s_1$	RFC-D3	RFC-D2	RFC-T2	RFC-T2	RFC-T2	RFC-Q2	4
$s_2$	CSTN-4096b	CSTN-4096b	CSTN-65Kc	TFT-65Ke	TFT-260Ke	TFT-65Ke	3
$s_3$	Li-Ion 600	Li-Ion 600	Li-Ion 780	Li-Ion 780	Li-Ion 780	Li-Ion 780	2
$s_4$	None	VGA30	VGA30	VGA30	VGA120	VGA120	2
$s_5$	None	None	Bluetooth 1	None	Bluetooth 1	Bluetooth 1	1
IC	0.000	0.525	0.003	0.166	0.000	0.007	–
Cost	75.15	83.25	112.59	108.50	138.76	144.03	–
<b>Average IC: 0.117    Total cost: 21.144(<math>\times 10^6</math>S\$)</b>							<b>12</b>

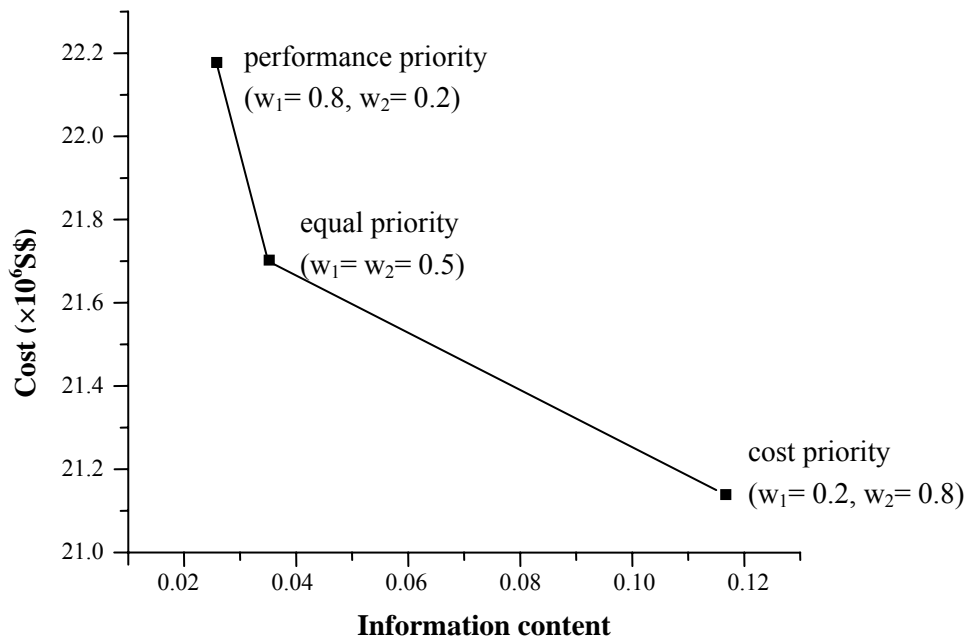


Figure 7.5 Objective functions of the solutions w.r.t. different priority strategies

### 7.2.3 Discussion

The cost of the product family based on cost priority was 21.144 ( $\times 10^6$  \$\$), which was about 5% lower than the cost based on performance priority ( $22.188 \times 10^6$  \$\$). The cost differences are attributable to the different levels of commonality, which can be roughly estimated as the number of different components implemented in the product family. With respect to the different priority strategies, the numbers of different components were 19 (performance priority), 14 (equal priority), and 12 (cost priority), respectively. This indicates that cost effectiveness has been achieved by increasing the commonality among the products in the family.

The average information content was used to measure the performance of the product family. The minimum average information content (0.026) was found when performance was given a higher priority ( $w_1=0.8$ ). The product family in this scenario could fulfill the design requirements to a higher extent. Based on Equation (5.5), the probability that the design requirements can be satisfied is 98%.

It should be noted that the average information content is used for the purpose of optimizing the entire product family. It may not reflect the performance of individual products. Instead, the information content of individual products indicates the performance of a particular product. For example, the information content of product  $P_2$  in Table 7.8 is 0.526, which means that the probability that the design requirement can be satisfied was only 70%. The main reason for this inferior performance is that

the KCs, namely, *MST* and *MTT*, were poorly satisfied. Studying the components of the product, a low capacity battery has been the cause of the poor *MST* and *MTT* performance. This gives the designer guidelines to improve the solution, e.g., replacing the low capacity batteries with high capacity ones.

This case study shows the design synthesis and evaluation process of product family design. The design problem was formulated as a configuration design problem and solved using the GA-based method. Cost and performance considerations can be effectively managed for product family design within the optimization framework. Information content is used as an integrated measurement of product performance. This is a useful improvement to product performance evaluation based on different performance criteria (e.g., the KCs). For example, if product performance was evaluated based on each and every engineering metrics (KCs), there would be a total of 42 objective functions (7 KCs multiplied by 6 products), based on different measurement units. It is difficult, if not impossible, to find an optimal tradeoff from among these measurements. Moreover, the estimation of performance based on information content can also be used to identify possible flaws of a design, and accordingly, provide guidelines for improvements.

### **7.3 Case Study II: TV Receiver Circuits Design**

This section presents a comparative study of the design of a family of TV receiver circuits. The purpose of this case study is to show (1) the effectiveness of the PFDR in

dealing with the product family configuration design problem, (2) the validity of the cost model, and (3) the efficacy of the ICA method in product performance evaluation. In particular, Fujita *et al.* (1999) proposed a modular method for product family configuration design. Using the same set of data, the PFDR method is used to solve the same problem. Results obtained from this method were compared to those from the benchmark method.

### 7.3.1 Settings

1. The target problem is the same as in the benchmark method, i.e., to design a family of six TV receiver circuits (Fujita *et al.*, 1999). Two scenarios were considered, namely, case 1 and case 2, in which the production volumes are different (Table 7.9).

Table 7.9 Product variety of TV sets

KCs		$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$
Picture size (inch)		14	21	36	14	21	36
Picture quality		Good	Better	Best	Normal	Good	Better
Audio level		Low	Medium	High	Low	Medium	Medium
Power supply voltage		100V	100V	100V	Multi	Multi	100V
Production volume	Case 1	36000	24000	12000	36000	24000	12000
	Case 2	24000	24000	24000	24000	24000	24000

2. The key characteristics of the product (called **feature indices** in the benchmark method) were kept the same (refer to the first column of Table 7.10).
3. The architecture of the product, as well as the functions and product modules, remained unchanged. In particular, seven module slots were reused. Based on the

original relationships between KCs, functions and modules, a correlation matrix was easily obtained (Table 7.10).

Table 7.10 Mapping from design requirements to components

KCs	M <sup>1</sup>	M <sup>2</sup>	M <sup>3</sup>	M <sup>4</sup>	M <sup>5</sup>	M <sup>6</sup>	M <sup>7</sup>
(Defaults)	1						
Picture size			1				
Picture quality		1		1	1		
Audio level						1	
Power supply voltage							1
M <sup>1</sup> - Tuner    M <sup>2</sup> - Picture signal processing    M <sup>3</sup> - Deflection circuit    M <sup>4</sup> - Color circuit M <sup>5</sup> - RGB driver    M <sup>6</sup> - Audio circuit    M <sup>7</sup> - Power supply							

4. To satisfy the product performance requirements, the PFDR method used the information content, instead of a set of design constraints, which was the case in the benchmark method. In order to do so, the capability indices of different components must be established. The capability indices were extracted from the attributes of the components. As an example, the capability indices of three deflection circuits ( $M^3$ ) are presented next.

Module  $M^3$  corresponds to the KC, *picture size*, which is an ordinal type KC and the default values are 14-inch, 21-inch and 36-inch. Three components  $m_1^3$ ,  $m_2^3$ ,  $m_3^3$  were available.  $m_1^3$  can be used for the 14-inch picture size only, while  $m_2^3$  can be used for both the 14- and 21-inch picture sizes.  $m_3^3$  can be used for all the three picture sizes. The capability indices of these three components were established as the pmfs shown in Figure 7.6. Similarly, the capability indices were established for



the other modules. It should be noted that, in the benchmark method, the ‘capacity constraint’ was used to deal with the interrelationships between modules. This constraint was retained in the PFDR method to define the feasible design space.

- For the product family cost, the PFDR method uses the cost model proposed in Chapter 6. To use this model, the cost elements and detailed data in the benchmark method was reformulated: first, the original cost elements were assigned to three levels; next, cost road-maps were established for each component based on the original data. The cost model in relation to the original cost elements in the benchmark method is shown in Table 7.11.

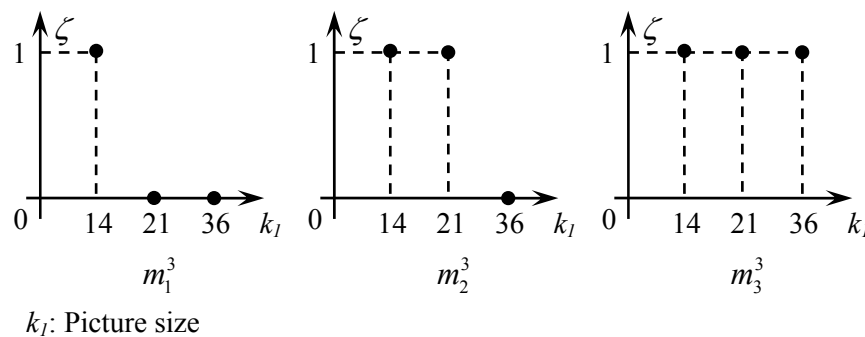


Figure 7.6 Component capability indices of three deflection circuit components

### 7.3.2 Solution generation and results

In the PFDR method, design synthesis is formulated as a multi-objective optimization problem, which was solved using the MOSGA method. Product performance was evaluated according to the information content. The algorithm generates the solution that minimized the product family cost and the average information content. The

information content of the optimal solutions is zero, which means that the design requirements can be safely satisfied. Product family cost results are shown in Table 7.12. Results of the benchmark method are reproduced.

Table 7.11 Cost model reformulation based on Fujita *et al.* (1999)

	<b>PFDR cost model</b>	<b>Original cost element</b>
Department level	$C_F = C_f^0 = 10,000,000$	$C_f^0 = 10,000,000$
Product level	$\sum C_P^i = C_v^a + C_f^P = 9,630,703$	$C_v^a = 645,703,$ $C_f^P = 8,985,000$
Component level	$C_{c(k)}^j = c_0 N_k^j \left( \frac{n_k^j}{n_0} \right)^{R_j}$ <p>where:  <math>c_0 = 1.8</math>, is the average cost per primitive.  <math>N_k^j</math> is the number of primitives in component <math>m_k^j</math>.  <math>n_k^j</math> is the number of component <math>m_k^j</math> in the product family.  <math>n_0 = 24000</math> is a base number of component.  <math>R_j</math> is the learning regression exponent, which is defined the same as in Equation (7.1).</p>	<p>Fixed cost:  <math>\sum C_f^{P(i)} ; \sum C_f^M ;</math></p> <p>Variable cost:  <math>C_v^m</math> - material cost;  <math>C_v^f</math> - facility cost.</p>

It should be noted that in the benchmark method, a primal solution is generated by the designer without considering product family optimality. The configuration of this primal solution and the resulting product family cost are compared with those of the optimal solutions, in case 1 and case 2 respectively. This strategy is also adopted in this research.

Table 7.12 Optimization results of product family cost

		Fujita's method			PFDR method		
		Primal	Optimal	diff. (%)	Primal	Optimal	diff. (%)
Case 1	# of diff. modules	20	11	-45%	20	11	-45%
	Total cost (10 <sup>6</sup> Yen)	133.112	130.279	-2.1%	133.317	131.219	-1.6%
Case 2	# of diff. modules	20	11	-45%	20	10	-50%
	Total cost (10 <sup>6</sup> Yen)	135.251	131.759	-2.6%	135.919	132.396	-2.4%

### 7.3.3 Discussion

The target problem in this case study is basically a configuration design problem. The benchmark method has formulated the design problem as a constraint-based, single objective optimization problem. Using the PFDR system, such a problem can be easily translated using the MOOP formulation and solved using the MOSGA method. The configurations of the optimal solutions based on the PFDR method and the benchmark method were similar to each other. In comparison to the primal solution, both optimal solutions reduce the number of components by 45 ~ 50%, achieving the same level of commonality. Since a higher commonality is key to cost reduction, the optimization method has effectively incorporated component commonality to reduce cost. Therefore, the PFDR method is effective in dealing with the product family configuration design problem.

The case study can partially verify the cost model developed in this research. As can be seen from the results, the total product family costs are very close using the PFDR method and the benchmark method in both cases. This is an indication that the cost model of the PFDR method has reflected the essential cost elements. In the PFDR

method, the cost reduction is 1.6% in case 1, and 2.4% in case 2. In the benchmark method, the optimal solutions achieve cost reductions of 2.1% and 2.6%, respectively. From the results, the benchmark method achieves a higher cost reduction. However, this is not an indication that the PFDR method is inferior in reducing cost. The differences can be attributed to the different formulations of the cost model. Although the cost model of the benchmark method is very comprehensive, it involves a number of parameters and coefficients that are difficult to be assigned accurately and reliably (Fujita, 2002). For example, the coefficients were assumed based on references rather than the actual production practices of the specific products. This is attributable to the ambiguity to forecast the design and manufacturing processes at the design and planning phase. On the other hand, the cost model developed in the PFDR method is constructed based on three levels of cost elements. The fixed cost and product development cost can be derived from the benchmark method without considering the details of the cost elements. The component cost is simulated using a cost road-map considering the learning effect. This has simplified cost estimation by alleviating the need to predefine a number of unpredictable cost parameters.

The case study also validates the efficacy of the ICA method. The performance requirements have been satisfied in both methods. In the benchmark method, this was achieved using three sets of design constraints which are problem specific. If the design requirements are changed, or the properties of the modules are modified, the constraints and optimization procedures have to be changed accordingly. Moreover, the

problem formulation is applicable to discrete KCs only. It is difficult to deal with a mixture of discrete and continuous KCs. In the PFDR method, performance is ensured through minimizing the information content. The design requirements and module properties can be changed without affecting the design synthesis and the evaluation process. Discrete and continuous KCs can be dealt with consistently in the ICA method. These have been made possible by the ICA method, which can consistently define the capability indices and compute the information content based on these indices. Thus, the PFDR method provides a more generic formulation to ensure product performance.

The major advantages of the PFDR method in comparison with the benchmark method are summarized in Table 7.13.

Table 7.13 Comparison of the PFDR method and the benchmark method

	<b>Benchmark method</b>	<b>PFDR method</b>	<b>Comment</b>
<b>Problem formulation</b>	Constraint-based single objective optimization.	Multiple objective optimization in a comprehensive design reuse framework.	PFDR provides a more generic formulation based on the design reuse methodology.
<b>Product performance</b>	Based on three sets of design constraints, which are problem specific.	The ICA method; Component capability indices and mapping from KCs to components derivable from existing cases.	The ICA method is applicable to different types of KCs. It presents standard procedures to compute the information content for performance evaluation.
<b>Cost model</b>	Based on detailed design and production information; Various parameters are assigned arbitrarily.	Based on a three-level cost model derivable from historical data.	Component cost road-map is used in the PFDR method to estimate cost. Relevant information can be obtained from existing designs.

### 7.4 Case Study III: Fan Filter Unit Design

The fan filter unit (FFU) is a key device in clean room products. It draws in air from outdoor space, removes the unwanted particles, and supplies clean and laminar airflow continuously into a clean room. Figure 7.7 illustrates the structure of a horizontally mounted FFU. The performance requirements of the FFU are rigorous and diverse with respect to different applications. In addition, a customer may have individual demands, such as a special size or a specific material. Therefore, the FFU manufacturers have to be able to provide customizable, low-cost products, while conforming to various industry standards. Traditionally, the FFU design has been largely dependent on a designer's expertise, with the support of computer-based or paper-based design catalogs. In comparison with the method presented in this research, the traditional method lacks a formalized design reuse foundation. Automated development of solution alternatives at the early design stage is rare; let alone a systematic estimation of the design superiority.

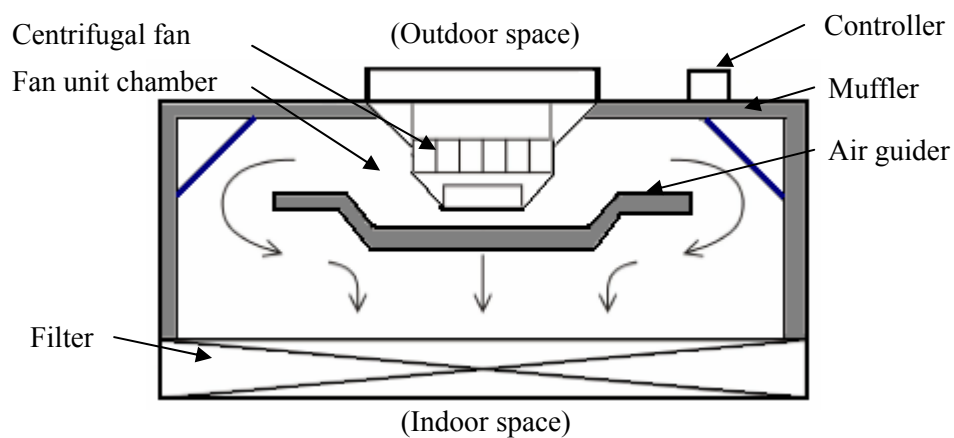


Figure 7.7 FFU structure and major components

The aim of the case study is to design FFU products using the design reuse method and show its effectiveness by comparing it with the traditional experience-based method. In particular, the case study will show the effectiveness of the PFDR system in organizing the overall design reuse process. The efficacy of the ICA method in product performance evaluation will be examined using a prototype product. To do so, first, the PFDR system was used to analyze existing product cases to establish a product platform that can accommodate a ranged set of design requirements. Next, a new FFU product was designed using the PFDR system. This task was also carried out by a design team following the traditional experience-based approach. The results were compared and the advantages of the PFDR method were discussed. It should be noted that in this case study, a single product was designed instead of a product family. This is to simplify the comparison between the PFDR method and the benchmark method. The PFDR system is capable of launching a product family *per se*.

#### **7.4.1 Establishment of product platform**

22 FFU product cases have been collected, modeled and stored in the database. The function structure of a typical FFU product is shown in Figure 7.8. The SOM method was used to generate the feature map (Figure 7.9), which facilitates the establishment of the FPA. The FPA consists of seven common functions (Figure 7.10). Next, ten KCs were formulated to cover the major customer needs (Table 7.14). The correlation between the KCs and the functions was established as  $TR_2$  (Table 7.15). A component catalog was built to include all the unique physical components which were assigned

to the component slots (Table 7.16).

The cost road-map and component capability indices were established based on the existing product cases. The corresponding attributes, such as the weight and cost of the components were included in the component catalog. Thus, a product platform was established for the FFU product.

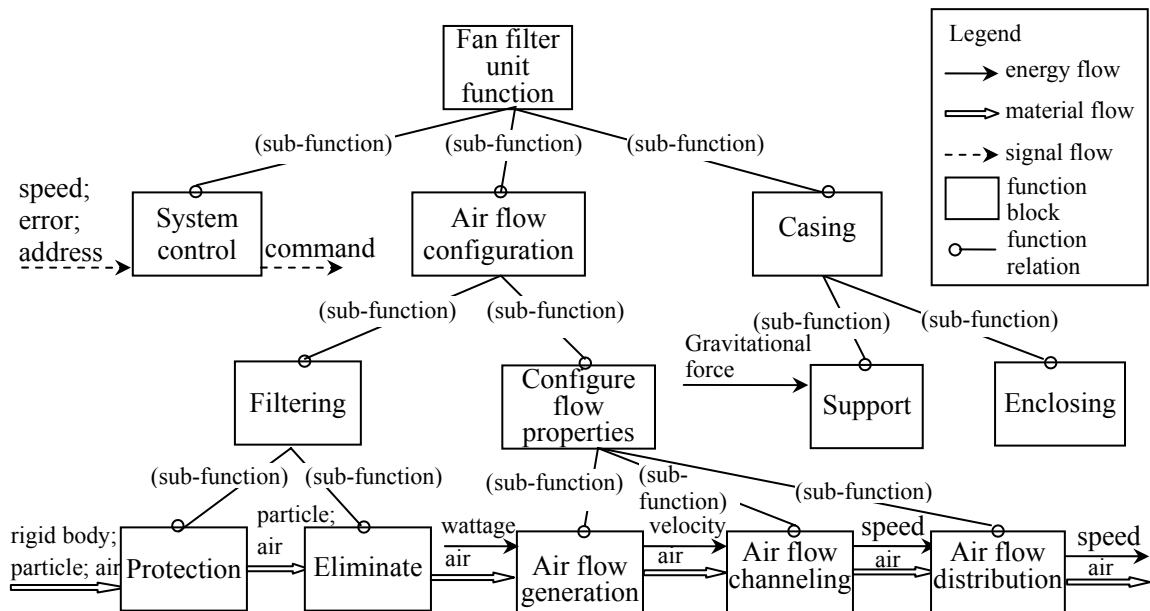


Figure 7.8 Function structure of FFU



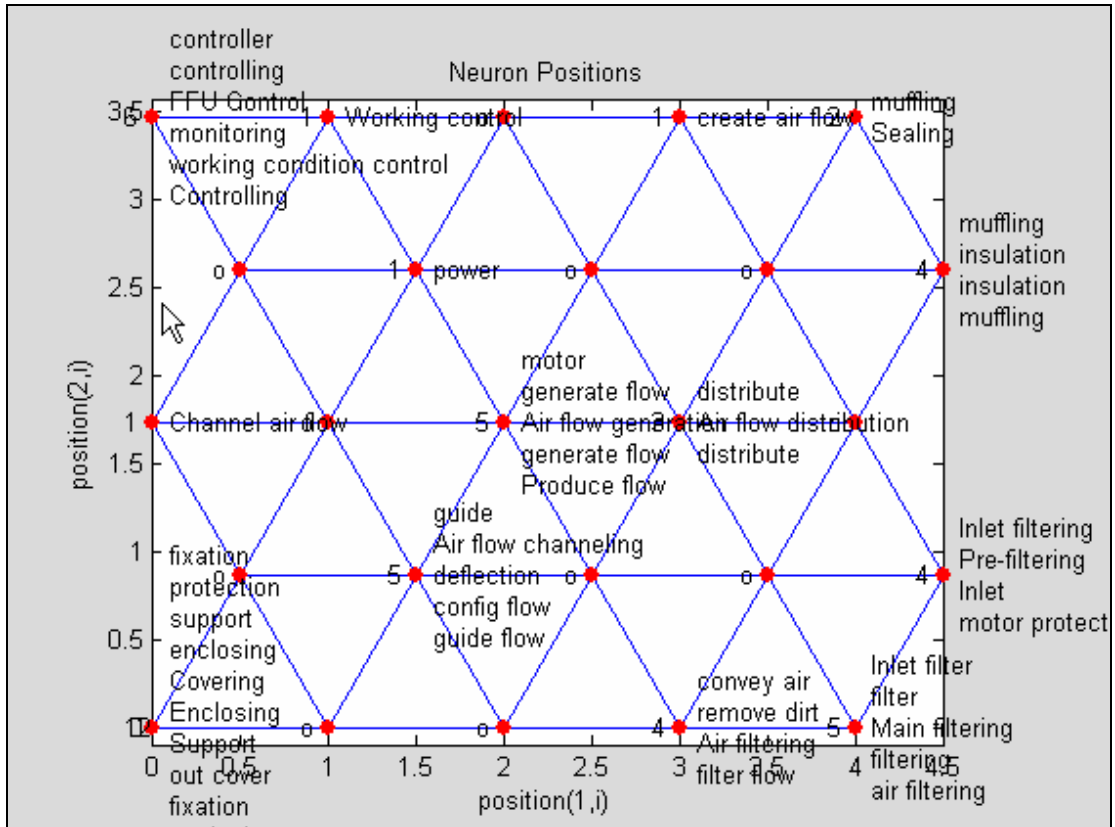


Figure 7.9 Feature map in the competitive layer (FFU)

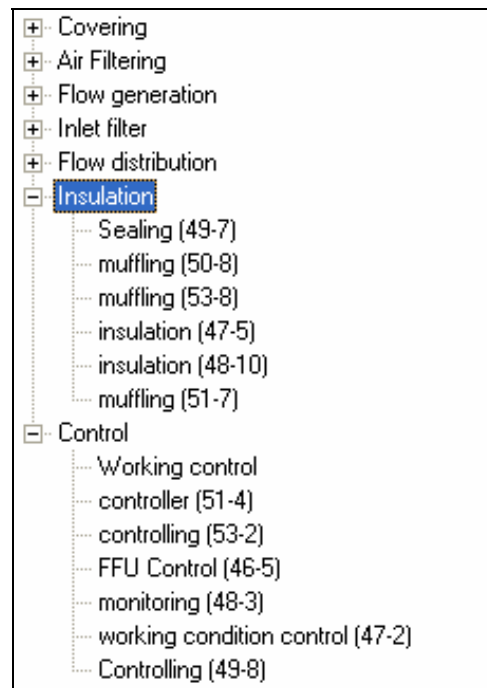


Figure 7.10 FPA of FFU products

Table 7.14 KCs of the FFU

KCs		Abbr.	Type	Unit	Default value
$k_1$	motor type	MT	categorical	–	1-AC-1PH; 3-AC-3PH; 2-EC
$k_2$	air quantity	AQ	real	m <sup>3</sup> /h	
$k_3$	air velocity	AV	real	m/s	
$k_4$	air uniformity	AU	real	%	
$k_5$	service cleanliness	SC	ordinal	–	1;10;100;1000;10000;100000
$k_6$	noise level	NL	real	dBA	
$k_7$	casing size	SC	real	–	
$k_8$	casing material	CM	categorical	–	aluminum; stainless steel; zinc-coated steel
$k_9$	vibration	VB	real	G	
$k_{10}$	mounting type	MO	categorical	–	1-Horizontal; 2-Vertical

Table 7.15 Correlation between KCs and functions ( $TR_2$ )

KCs		$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$k_1$	MT	1	0	0	0	0	0	0
$k_2$	AQ	1	0	0	1	0	0	0
$k_3$	AV	1	0	0	1	0	0	0
$k_4$	AU	0	0	1	0	0	1	0
$k_5$	SC	0	0	1	0	0	0	0
$k_6$	NL	1	0	0	0	0	1	1
$k_7$	SC	0	0	0	1	0	0	0
$k_8$	CM	0	0	0	1	0	0	0
$k_9$	VB	1	0	0	0	0	0	0
$k_{10}$	MO	0	0	0	1	1	0	0

$f_1$ : Flow generation;  $f_2$ : Control;  $f_3$ : Air filtering;  $f_4$ : Casing;  $f_5$ : Inlet cone;  
 $f_6$ : Flow distribution;  $f_7$ : Insulation

Table 7.16 Function and component slot

Function		Component slot		Number of components
$f_1$	Flow generation	$s_1$	Motor/Blower	11
$f_2$	Control	$s_2$	Controller	3
$f_3$	Air filtering	$s_3$	Filter	9
$f_4$	Casing	$s_4$	Casing	11
$f_5$	Inlet cone	$s_5$	Motor support/Inlet cone	6
$f_6$	Flow distribution	$s_6$	Air guiders	3
$f_7$	Insulation	$s_7$	Muffler	3

### 7.4.2 Configuration design of FFU using two methods

To study the effectiveness of the PFDR method, parallel designs were carried out by two groups of designers with respect to the same set of design requirements (Table 7.17). Group A carried out the design based on experience. Group B performed the task using the PFDR system. Both groups consist of two designers, namely, a product planner who developed the configuration of the product, and a technical engineer who was responsible of estimating the technical feasibility. Designers in the two groups are at the same level of experience, i.e., the average years of professional experience were equivalent (3.5 years). The product planner in Group B was given a short training course to use the PFDR system. The two groups generated the product configurations independently. For clarity, the product configuration produced by Group A is called Product A, denoted as,  $P_A$ , and that produced by Group B is called Product B, denoted as  $P_B$ .

Table 7.17 Design requirements of the FFU product

KCs	Value
motor type	AC-1PH
air quantity	$1375m^3/h \pm 10\%$
air velocity	$0.42m/s \pm 10\%$
air uniformity	$\leq 15\%$
service cleanliness	Class1000
noise level	$\leq 50dBA$
casing material	aluminum
casing size	3×4
vibration	$< 0.025G$
mounting type	Horizontal

### 7.4.2.1 Experience-based design

The experience-based method usually involves procedural selection and combination of the design components. Each component is selected or designed to achieve a set of performance criteria, as is defined by the design requirements. The designers rely heavily on individual/team experience, documentation and simulation results. Therefore, personal preferences significantly influence the efficiency and validity of the design. Since the components are determined by a number of factors, it is not easy to manage them simultaneously. Modifications and rework are often inevitable.

The configuration of  $P_A$  is shown in Table 7.18. It should be noted that the division of the design components for  $P_A$  is different from that for  $P_B$ . However, for convenience of comparison, the components have been rearranged into an identical sequence in this thesis.

Table 7.18  $P_A$  – product configuration generated by the experience-based method

Slot	Component	Cost (S\$)
Motor	R4E 310-AP20-01	130
Controller	S-1	10
Filter	HEPA filter 70	200
Casing	aluminum casing (redesigned)	220
Inlet cone	Square punched hole	10
Air guiders	3-set with array holes	15
Insulation	Rock-wool	5
<b>Sum</b>	–	<b>590</b>

### 7.4.2.2 PFDR for FFU design

Using the PFDR method, solution synthesis and the evaluation process were carried out using the MOSGA method. The product configuration generated by the MOSGA is shown in Table 7.19.

Table 7.19  $P_B$  – product configuration generated by the PFDR method

Slot	Component	Cost (S\$)
Motor	R4E 355-AL02-01	145
Controller	SLC-S1	10
Filter	HEPA 70+	180
Casing	AL shaft-standard-A3-4	186
Inlet cone	Circular 1	12
Air guiders	3-set with array holes	15
Insulation	Rock-wool and fiber glass	5
<b>Sum</b>	–	<b>553</b>

### 7.4.2.3 Comparison of product configurations

The configurations generated by the two methods differ in various aspects. The following discussions focus on two major components, namely, the motor and the casing.

- **Motor**

The diameters of the two motors are different. The diameter of Motor A is 310mm and that of Motor B is 355mm. Different diameters have led to different capacities of air flow generation. At normal working modes, the air flow volume of Motor A is 1185~1380m<sup>3</sup>/h, and the air flow volume of Motor B is 1449~1848m<sup>3</sup>/h. The air flow

capacity of the motor determines the air quantity of the FFU. However, the air quantity of the FFU is less than the rated air flow volume of the motor because of the pressure loss caused by the air channel and the filters. In general, the air quantity of the FFU is 85~90% of that of the air flow volume of the motor. To achieve the required air quantity ( $1375m^3/h \pm 10\%$ ), Motor A has to work at a high speed near its maximum capacity, which is not favorable, while Motor B could achieve the goal at its normal working mode.

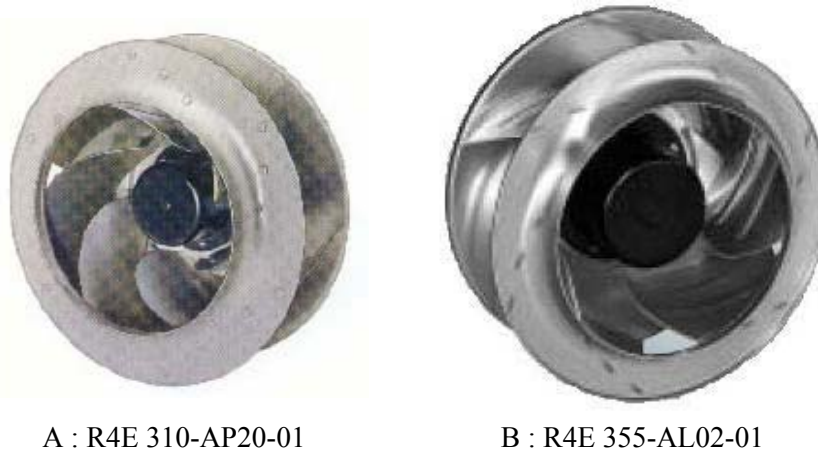


Figure 7.11 Motors used in  $P_A$  and  $P_B$

- **Casing**

A casing provides the function of enclosing and supporting the other components. A typical structure of a casing is shown in Figure 7.12. The motor was fixed on a supporting plate, which is in turn underpinned by a reinforced plate. The reinforced plate is fixed on the casing walls using four nuts and bolts around four corners.  $P_B$  used this standard casing structure.

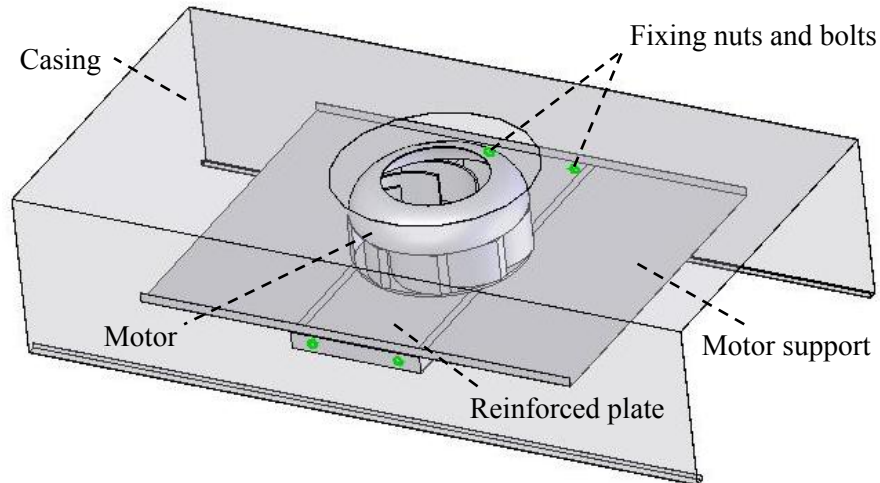


Figure 7.12 Standard casing structure ( $P_B$ )

Since this structure has been used and tested in many cases, it is very safe and well-documented. Only minor changes are needed to accommodate new design requirements. Therefore, the casing for  $P_B$  was designed quickly and efficiently. Consequently, the manufacturing and assembly become routine and cost effective.

One drawback of this casing structure is that the motor is rigidly fixed to the casing, i.e., the connection between the motor and the casing is very stiff. Noise and vibration can be transmitted from the motor to the casing and other components. This is not a big concern in  $P_B$  because the motor works in normal mode and the noise and vibration level is low. However, it poses a problem for  $P_A$  as is discussed next.

In  $P_A$ , since the air flow volume generated by the motor is barely sufficient, the motor has to work at an extreme high-speed mode. Noise and vibration would be so high that the standard casing for motor fixation is not applicable. A new structure was designed,

which is shown in Figure 7.13. The motor is supported by a bottom plate connected to the top cover plate using four long bolts. The entire supporting sub-assembly is, in turn, suspended on the upper plane of the casing.

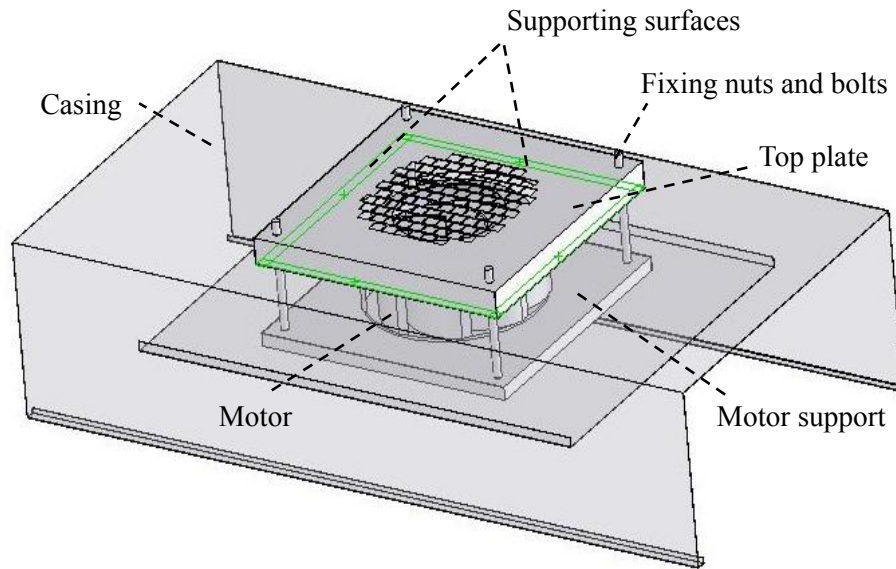


Figure 7.13 Redesigned casing structure ( $P_A$ )

According to the experiments, this type of motor fixation provides more flexibility in connection. It helps to reduce noise and vibration. Despite these good features, the casing has to be redesigned to incorporate the changes, besides the modifications to the other components. These increase the overall cost and cause additional design time.

#### 7.4.2.4 Comparison of product performance

The performance of  $P_A$  were tested and presented in Table 7.20. It was found that most performance requirements were satisfied, except that the air quantity only marginally reaches the target level. In particular, the air quantity ( $1235m^3/h$ ) is approximately



equal to the lower limit of the requirement ( $0.9 \times 1375 = 1237.5 m^3/h$ ). As has been discussed earlier, the insufficient air quantity is mainly attributable to the choice of the motor. In order to accommodate the desired air quantity, the motor has to work at a high rotational speed near its maximum capacity.

Table 7.20 Performance of  $P_A$ 

KC	Value
motor type	AC-1PH
air quantity	$1235 m^3/h$
air velocity	$0.41 m/s$
air uniformity	15%
service cleanliness	1000
noise level	$50.0 dBA$
casing material	aluminum
casing size	$3 \times 4$
Vibration	$0.025 G$
mounting type	Horizontal

For  $P_B$ , information content was used to evaluate product performance. The details to compute the information content of the solution is presented in Table 7.21. As can be seen from the individual information content values, information content with respect to the six design requirements ( $MT$ ,  $SC$ ,  $NL$ ,  $CS$ ,  $CM$ ,  $MO$ ) were 0. According to Equation (5.5), it means that the probability that these design requirements can be safely satisfied is 1. Three design requirements (w.r.t.  $AQ$ ,  $AV$  and  $AU$ ) can be satisfied with high probability. In particular, the probabilities of meeting these performance targets are all above 99%. These can be derived from the information content values: 0.003, 0.007, and 0.001. The largest information content was detected in the KC  $VB$ ,

0.044, indicating a 97% probability that the vibration can be kept below the desired level. Since the vibration feature is mainly dependent on the attributes of the motor, which can be adjusted slightly to change the vibration level, this KC can be satisfied without major changes in product configuration.

The pseudo product information content is 0.055, which is the summation of the information content with respect to the individual KCs. From this, it is estimated that the probability that all design requirements can be satisfied is 96%. From the analysis of the individual information content, it is expected that the product performance can be achieved by reusing the components.

Table 7.21 Computation of information content

KC	Design range	System range	Information content
		$\zeta_1 = \begin{cases} 1, & MT=AC-1PH; \\ 0, & MT=AC-3PH EC. \end{cases}$	
motor type ( <i>MT</i> )	AC-1PH		0.000
		Normal distribution: $\mu=1449, \sigma=22$	
air quantity ( <i>AQ</i> )	[1237.5, 1512.5]		0.003
air velocity ( <i>AV</i> )	[0.378, 0.462]	Normal distribution: $\mu=0.42, \sigma=0.015$	0.007
air uniformity ( <i>AU</i> )	$\leq 15\%$	Normal distribution: $\mu=0.11, \sigma=0.012$	0.001
service cleanliness ( <i>SC</i> ) 1000		$\zeta_5 = \begin{cases} 1, & SC=1000 10000; \\ 0, & SC=1 10 100. \end{cases}$	0.000
noise level ( <i>NL</i> )	50	$\zeta_6 = \begin{cases} 0, & NL \leq 47; \\ 0.5(NL - 47), & 47 < NL \leq 49; \\ 1, & NL > 49. \end{cases}$	0.000
casing size ( <i>CS</i> )	3×4	$\zeta_7 = \begin{cases} 1, & CS=3 \times 4; \\ 0, & CS=2 \times 4 4 \times 4. \end{cases}$	0.000
casing material ( <i>CM</i> )	aluminum	$\zeta_8 = \begin{cases} 1, & CM=Aluminum \\ 0, & \text{others.} \end{cases}$	0.000
vibration ( <i>VB</i> )	$\leq 0.025$	Normal distribution: $\mu=0.0235, \sigma=0.0008$	0.044
mounting type ( <i>MO</i> )	Horizontal	$\zeta_{10} = \begin{cases} 1, & MO=Horizontal; \\ 0, & MO=Vertical. \end{cases}$	0.000

A prototype product of  $P_B$  was produced and the performance was tested. This is to examine whether the product performance agrees with the prediction of the ICA method. The results are presented in Table 7.22. As can be seen from the table, the performance of  $MT$ ,  $SC$ ,  $NL$ ,  $CS$ ,  $CM$ ,  $MO$  has been satisfied. This agrees with the ICA prediction where all the information contents are '0'. The performance of  $AQ$ ,  $AV$ , and  $AU$  is slightly different from the design targets, which agrees with the estimation of the information content. These results show that information content is effective in the evaluation of the product performance. An exception occurred with respect to the prediction of  $VB$ . While the information content was 0.044, indicating a 97% probability that the vibration can be kept below the desired level, the prototype product has satisfied the performance requirement. The reason for this discrepancy is that the motor attribute, namely, the vibration level, can be adjusted when the prototype product was being built. Therefore, before the performance test is carried out, the  $VB$  has been adjusted to the desired level. This situation gives rise to an important implication of using the ICA method. Information content has statistical significance for predicting product performance. However, it does not consider specific changes to the product that may influence the product performance. When such specific changes are present, information content could not be used as an accurate estimation of the product performance.

#### **7.4.2.5 Comparison of product cost**

The overall cost of  $P_B$  is S\$553, about 6.7% lower than the cost of  $P_A$  (S\$590). The

major differences are found in the motor, filter, and casing.  $P_A$  uses a cheaper motor to reduce product cost. However, this hardly proves to be effective because it worsens the noise and air quantity properties, which has elicited the redesign of the casing to counteract such effects. This directly increases the cost of the casing. Moreover, to achieve the desired air uniformity,  $P_A$  uses a filter with better uniformity properties. This filter is more expensive than the one used in  $P_B$ . All these factors contribute to the higher cost of  $P_A$ .

Table 7.22 Performance of  $P_B$ 

KC	Value
motor type	AC-1PH
air quantity	$1385m^3/h$
air velocity	$0.425m/s$
air uniformity	13%
service cleanliness	1000
noise level	$49.2dBA$
casing material	aluminum
casing size	$3 \times 4$
vibration	$0.025G$
mounting type	Horizontal

### 7.4.3 Discussion

The PFDR method was compared with the experience-based design method in this case study. From the design process and results, it is shown that:

- (1) The PFDR method can consistently manage the design reuse process. It accommodates the requirements of product case modeling, product architecture building, and automated generation of a product configuration through an efficient

exploration of the design space. In comparison with the traditional experience-based approach, product information can be retained and retrieved more effectively.

- (2) The characteristics of the new product can be better predicted. This is because product configuration is generated based on existing components. The PFDR method provides systematic ways to estimate the product characteristics based on existing components. The capability indices of these components were established using statistical tools. As long as the initial product information is collected and formulated in the design reuse system, new products can be easily evaluated.
- (3) The efficacy of the information content in predicting product performance has been examined using a prototype product. It is shown that information content gives generally valid prediction of product performance. However, specific changes to the products may impair the efficacy of information content in product performance evaluation.
- (4) The design reuse method is not a replacement of the experience or expertise of the designers. Design reuse is suitable for variant design where the solutions have been used in various forms in previous designs. It is expected that human intelligence is still indispensable to generate innovative solutions. For example, in the FFU case study, the new casing structure generated by Group A shows significant improvements in reducing noise and vibration. Although the design and manufacturing cost is higher, it is a promising, innovative solution that can be improved in future designs.

## 7.5 Summary

The design reuse methodology was implemented using a prototype product family design reuse system. Three case studies were presented to demonstrate the effectiveness of the PFDR method. First, the design of the cellular phone product family shows that the PFDR method can explore the designs space and effectively manage the trade-offs between the performance and cost of a product. Next, the PFDR method was benchmarked with a modular design method in the case of the TV receiver circuits. The PFDR method provides a more consistent problem formulation and solution generation. Finally, the method was applied to the design of an industrial product, namely, the FFU. The design results were compared with those obtained using the traditional experience-based design method. The PFDR method can generate solutions that are superior in both cost and performance.

# Chapter 8 CONCLUSIONS AND FUTURE WORK

## 8.1 Conclusions

Product family design is a proven method to increase product variety while maintaining production efficiency. Information deficiency and uncertainty is a major hindrance to product family design at the conceptual stage. Decision-making in such a context presents a big challenge to the designers. Design reuse is a promising approach to solve this problem. However, current practices in design reuse have been inadequate to provide a complete design process model. Accordingly, the following research issues have been identified in Chapter 2: (1) the development of a comprehensive design process model, (2) the establishment of a function-based product architecture, (3) the adoption of information content as a uniform metric, and development of logical procedures to compute the information content, and (4) the formulation of the design synthesis problem and application of efficient optimization algorithms to solve the design synthesis problem. To address these issues, this research proposes a design reuse methodology to enhance the efficiency of product family design. The following conclusions can be made.

A comprehensive PFDR process model was developed to manage information modeling, information analysis, and design synthesis and evaluation. Information



modeling is based on a product information representation scheme that incorporates the multiple facets of product information. Formal representation of product functions with the support of taxonomies is an important element to enhance the representation rigor. A set of knowledge extraction operators are explicitly defined to reformulate the raw product data into reusable forms. The MOSGA method is used to automate the design synthesis process, which enhances the efficiency of solution generation. In comparison with traditional methods, such as case-based reasoning, catalog-based design, and various design repositories, the PFDR framework presents a holistic design reuse process model to facilitate product family design. Within this framework, the most important stages of product family design are logically connected. Each stage is supported by techniques to automate the process to a certain extent. For example, information modeling has a function core that is based on a comprehensive data structure and an established taxonomy. Design synthesis is formulated as a multiple objective optimization problem, and solved using GA-based search methods to achieve an efficient exploration of the design space. These enhance the reusability of the product information, and facilitate intelligent design reuse. Thus, information deficiency and uncertainty can be alleviated to a certain extent.

Concerning the building of function-based product architecture, intelligent analysis tools are expected to alleviate the human labor and enhance the efficiency of product architecture building. The product architecture design problem was formulated as a function clustering problem (Chapter 4). Next, the neural networks technique, namely,

the SOM method was used to facilitate the building of function-based product architectures. This method is based on formal function modeling techniques, such as the KEV representation and taxonomy. The method distinguishes itself from the traditional methods through the adoption of unsupervised learning techniques. Through the results of the test example in Chapter 4, it is demonstrated that the SOM method has achieved (1) an expedition of the process, (2) an alleviation of human labor, (3) the determination of useful initial knowledge and patterns of the architecture, and (4) an analysis of the data from a perspective other than empirical observation. These signify significant improvements to traditional methods that rely heavily on human expertise or heuristic rules.

Moreover, the research developed a set of knowledge extraction operators to capture the relevant information from the raw product data. For example, the reusability of product components can be assessed by developing capability indices for the design components, which, in turn, provide a useful way to estimate the product performance during the design synthesis and evaluation stage. These knowledge extraction operators can transform raw product information into reusable forms that constitute a product platform for product family design.

Information content is adopted as a dimensionless, uniform metric that incorporates diverse measures of performance criteria. Product performance can be evaluated and ensured by minimizing the information content. In comparison to other evaluation

methods, such as the constraint-based methods (Case study II, Section 7.3), the ICA method provides a more generic foundation for performance evaluation. This not only helps to ensure product quality, but also simplifies the formulation of the design synthesis problem, and enhances computational support. Moreover, considering the inadequacy of existing approaches in computing the information content, the ICA method proposed in this research defines the logical steps to compute the information content (Section 5.2). Most notably, it provides mechanisms to establish the capability indices for product components, which can be used to establish the system ranges of products. As is shown in case study III (Chapter 7), the system ranges can be formulated based on existing product information. Thus, this method naturally follows the spirit of design reuse. Moreover, through a comparison of the product performance predicted using the ICA method and that tested from a prototype product, product performance can be effectively predicted using the information content.

Finally, the product family configuration design problem is formulated as a multiple objective optimization problem. Cost and product performance are used as the two objective functions, and the reusable product components are the design parameters. A GA-based search algorithm, namely, MOSGA is adopted to solve the configuration design problem. The capacity of MOSGA has been demonstrated through case study III in Chapter 7, where the algorithm outperforms the experience-based method by finding a solution that excels in both cost and product performance. It can be

concluded that MOSGA can effectively explore the design space and find the Pareto-optimal solution set.

Based on the above discussions, the contributions of this research include:

- (1) Proposing the PFDR methodology which provides a relatively complete process model for product family design reuse.
- (2) Applying the SOM method to automate the building of function-based product architectures.
- (3) Applying the information content as a uniform, dimensionless measurement of the performance of products.
- (4) Developing the ICA process to derive system ranges and evaluate product performances.

## **8.2 Future Work**

A few limitations of the proposed method are discussed next. Future work can be carried out to overcome these limitations.

The representation of product function using taxonomies restricts flexibility. The SOM method for the establishment of the PFA depends on a formal representation of the product functions. Although the information model strives to capture the product information in a comprehensive way, the use of a formal function structure and taxonomy is restrictive and inevitably reduces flexibility. It is desirable to develop

knowledge extraction techniques based on a more flexible representation of the product information.

Another limitation of the research is related to the validation of the SOM method. Although the effectiveness of the SOM method has been tested using the test example in Chapter 4 and the case study in Chapter 7, the validation is inadequate in terms of the number and variety of product cases, and the complexity of the products. To further validate its capacity and efficacy, it is desirable to test the method with more complex products, such as automobiles, aircrafts, power tools, etc. Moreover, although the SOM method uses unsupervised learning, the ultimate formation of the product architecture still requires some human intelligence. Given the same set of data, different designers can arrive at different product architectures, leading to different effectiveness of product information management. It is proposed that future work be carried out to test the effectiveness of the method by studying the outcome generated by designers with different knowledge backgrounds. Such a testing process is important to determine whether the method is applicable to a broader range of applications.

The calculation of the information content presents another limitation of the research. In this thesis, the summation of the individual information content with respect to each and every KC is taken as the information content of a product. However, this is only applicable when the KCs are independent of each other. When this condition is not met, errors may occur, which may affect the validity of subsequent design evaluation.

Therefore, the summation of the individual information content is only an approximation of the information content of the product. Future research can be carried out to incorporate the correlations between the KCs and compute the information content using the joint probability formulation (Equation 5.2).

Finally, the dimensionless value of the information content may be less meaningful to a human designer. It is not as straightforward in meaning as the other criteria, such as product cost or weight. A designer may have to be accustomed to the principles of probability and think of information content in terms of the probabilities before he/she can utilize the method. It may be beneficial to convert the information content into more intuitive forms to increase clarity and enhance communication between designers. Some efforts have been made in this respect, e.g., the design customizability index which is a metric based on information content to measure the cost-effectiveness of a customization feature (Jiao and Tseng, 2004), and the design capability index which is used to assess the capability of a product family to satisfy diverse design requirements (Simpson *et al.*, 1997). However, these converted forms are usually problem specific, and hence, have not been widely adopted in literature. In contrast, information content has a simpler mathematical foundation, making it a more generic metric. Therefore, this research used information content in its original form.

## **PUBLICATIONS FROM THIS THESIS**

Xu, Q.L., Ong, S.K. and Nee, A.Y.C., 2006, Function-based Design Synthesis Approach to Design Reuse, *Research in Engineering Design*, **17**(1), pp. 27–44.

Ong, S.K., Xu, Q.L. and Nee, A.Y.C., 2006, Design Reuse Methodology for Product Family Design, *Annals of the CIRP*, **55**(1), pp. 161–164.

Xu, Q.L., Ong, S.K. and Nee, A.Y.C., 2005, Design Synthesis and Evaluation in a Design Reuse System, In: *Proceedings of 2005 ASME International Mechanical Engineering Congress and Exposition*, Paper No. IMECE2005-79463, Orlando, Florida, November.

Xu, Q.L., Ong, S.K. and Nee, A.Y.C., 2006, Evaluation of Product Performance in Design Reuse, *International Journal of Production Research*, accepted for publication.

---

## REFERENCES

- Andersson, J. and Wallace, D., 2002, Pareto Optimization Using the Struggle Genetic Crowding Algorithm, *Engineering Optimization*, **34**, pp. 623–644.
- Bahrami, A., 1994, Routine Design with Information Content and Fuzzy Quality Function Deployment, *Journal of Intelligent Manufacturing*, **5**(4), pp. 203–210.
- Baxter, J., Juster, N. and de Pennington, A., 1994, A Functional Data Model for Assemblies Used to Verify Product Design Specifications, In: *Proceedings of the Institution for Mechanical Engineers, Part B – Journal of Engineering Manufacture*, **208**, pp. 235–244.
- Bobrow, D., Falkenhainer, B., Farquhar, A., Fikes, R., Forbus, K.D., Gruber, T.R., Iwasaki, Y. and Kuipers, B.J., 1996, A Compositional Modeling Language, In: *Proceedings of the 10th International Workshop on Qualitative Reasoning*, Menlo Park, CA., May, AAAI Press, pp. 12–21.
- Campbell, M., Cagan, J. and Kotovsky, K., 1999, A-Design: an Agent-based Approach to Conceptual Design in a Dynamic Environment, *Research in Engineering Design*, **11**, pp. 172–192.
- Chakrabarti, A. and Bligh, T., 1996, An Approach to Functional Synthesis of Mechanical Design Concepts: Theory, Applications, and Emerging Research Issues, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**, pp. 313–331.
- Chandrasekaran, B., Goel, A. and Iwasaki, Y., 1993, Functional Representation as Design Rationale, *IEEE Computer*, **26**(1), pp. 48–56.



- Chen, W., Allen J.K., Mavris, D.N. and Mistree, F., 1996, A Concept Exploration Method for Determining Robust Top-Level Specifications, *Engineering Optimization*, **26**, pp. 137–158.
- Chidambaram, B. and Agogino, A.M., 1999, Catalog-based Customization, In: *Proceedings of 1999 ASME Design Engineering Technical Conferences- Design Automation Conference*, Paper No. DETC99/DAC-8675, Las Vegas, Nevada, September.
- Clausing, D., 1994, *Total Quality Development: a Step-by-Step Guide to World Class Concurrent Engineering*, New York: ASME Press.
- Corbett, B. and Rosen, D.W., 2004, A Configuration Design Based Method for Platform Commonization for Product Families, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **18**, pp.21–39.
- Cooper, R. and Kaplan, R.S., 1991, *The Design of Cost Management Systems*, Prentice-Hall, NJ: Englewood Cliffs.
- Counsell, J., Porter, I., Dawson, D. and Duffy, M., 1999, Schemebuilder: Computer Aided Knowledge Based Design of Mechatronic Systems, *Assembly and Automation*, **19**(2), pp. 129–138.
- D’Souza, B. and Simpson, T.W., 2003, A Genetic Algorithm Based Method for Product Family Design Optimization, *Engineering Optimization*, **35**(1), pp. 1–18.
- Dahmus, J.B., Gongzalez-Zugasti, J.P. and Otto, K.N., 2000, Modular Product Architecture, In: *Proceedings of the 2000 ASME Design Theory and Methodology Conference*, Paper No. DETC2000/DTM-14565, Baltimore, MD.
- Deb, K., 2001, *Multi-Objective Optimization Using Evolutionary Algorithms*, New York: John Wiley & Sons.

- Dong, Q. and Whitney, D.E., 2001, Designing a Requirement Driven Product Development Process, In: Proceedings of ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences, Paper No. DETC2001/DTM-21682, Pittsburgh, Pennsylvania, September.
- Du, X., Jiao, J. and Tseng, M.M., 2001, Architecture of Product Family: Fundamentals and Methodology, *Concurrent Engineering: Research and Applications*, **9**(4), pp. 309–325.
- Du, X., Jiao, J. and Tseng, M.M., 2002a, Graph Grammar Based Product Family Modelling, *Concurrent Engineering: Research and Applications*, **10**(2), pp. 113–128.
- Du, X., Jiao, J. and Tseng, M.M., 2002b, Product Family Modeling and Design Support: An Approach Based on Graph Rewriting Systems, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **16**(2), pp. 103–120.
- Duffy, A.H.B. and Ferns, A.F., 1999, An Analysis of Design Reuse Benefits, In: Proceedings of the ICED 99 Conference, Lindemann, U., Birkhofer, H., Meerkamm, H. and Vajna, S., (eds.), Technische Universität München, 1999, pp. 799–804.
- Duffy, S.M., Duffy, A.H.B. and MacCallum, K.J., 1995, A Design Reuse Process Model, In: Proceedings of the International Conference on Engineering Design (ICED95), Prague, August, Heurista Zurich, pp. 490–495.
- Duffy, A.H.B., Persidis, A. and MacCallum, K.J., 1996, NODES: A Numerical and Object Based Modeling System for Conceptual Engineering Design, *Knowledge-Based Systems*, **9**, pp. 183–206.
- Erixon, G., 1996, Design for Modularity, In: Design for X – Concurrent Engineering

- Imperatives, Huang, G.Q. (ed.), pp. 356–379, New York: Chapman & Hall.
- Ericsson, A. and Erixon, G., 1999, *Controlling Design Variants: Modular Product Platforms*, New York: ASME Press.
- Erens, F.J., McKay, A. and Bloor, S., 1994, Product Modeling Using Multiple Levels of Abstraction Instances as Types, *Computers in Industry*, **24**(1), pp. 17–28.
- Felfernig, A., Friedrich, G. and Jannach, D., 2001, Conceptual Modeling for Configuration of Mass-Customizable Products, *Artificial Intelligence in Engineering*, **15**(2), pp. 165–176.
- Fonseca, C. and Fleming, P., 1998, Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation, *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems & Humans*, **28**(1), pp. 26–37.
- Fujita, K., 2002, Product Variety Optimization under Modular Structure, *Computer-Aided Design*, **34**, pp. 953–965.
- Fujita, K., Akagi, S., Yoneda, T. and Ishikawa, M., 1998, Simultaneous Optimization of Product Family Sharing System Structure and Configuration, In: *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, Paper No. DETC98/DFM-5722, Atlanta, Georgia, September.
- Fujita, K. and Yoshida, H., 2001, Product Variety Optimization: Simultaneous Optimization of Module Combination and Module Attributes, In: *Proceedings of ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences*, Paper No. DETC2001/DAC-21058, Pittsburgh, Penn., September.
- Fujita, K. and Yoshioka, S., 2003, *Optimal Design Methodology of Common*

- Components for a Class of Products: Its Foundations and Promise, In: Proceedings of ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences, Paper No. DETC2003/DAC-48718, Chicago, Illinois, September.
- Fujita, K., Sakaguchi, H., Akagi, S. and Yoneda, T., 1999, Product Variety Development and its Optimization under Modular Architecture and Module Commonalization, In: Proceedings of the 1999 ASME Design Engineering Technical Conferences, Paper No. DETC99/DFM-8923, Las Vegas, Nevada, September.
- Gero, J.S., 1990, Design Prototypes: A Knowledge Representation Schema for Design, *AI Magazine*, **11**(4), pp. 26–36.
- Gonzalez-Zugasti, J.P. and Otto, K.N., 2000, Modular Platform-based Product Family Design, In: Proceedings of the 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Paper No. DETC2000/DAC-14238, Baltimore, Maryland, September.
- Gonzalez-Zugasti, J.P., Otto, K.N. and Baker, J.D., 2000, A Method for Architecturing Product Platforms, *Research in Engineering Design*, **12**, pp. 61–72.
- Gonzalez-Zugasti, J.P., Otto, K.N. and Baker, J.D., 2001, Assessing Value in Platformed Product Family Design, *Research in Engineering Design*, **13**, pp. 30–41.
- Gorti, S.R. and Sriram, R.D., 1996, From Symbol to Form: A Framework for Conceptual Design, *Computer-Aided Design*, **28**(11), pp. 853–870.
- Grante, C. and Andersson, J., 2003, A Method for Evaluating Functional Content in Mechatronic Systems, *Research in Engineering Design*, **14**, pp. 224–235.

- Gu, P. and Sosale, S., 1999, Product Modularization for Life Cycle Engineering, Robotics and Computer Integrated Manufacturing, **15**, pp. 387–401.
- Hata, T., Kimura, F. and Suzuki, H., 1997, Product Life Cycle Design Based on Deterioration Simulation, In: Life Cycle Networks, 4th CIRP International Seminar on Life Cycle Engineering, Krause, F.-L. and Seliger, G., (eds.), pp. 59–68, 1997, London: Chapman & Hall.
- Haykin, S., 1999, Neural Networks: A Comprehensive Foundation, 2nd ed., Upper Saddle River, NJ: Prentice Hall.
- Hernandez, G., Allen, J.K. and Mistree, F., 2002, Design of Hierarchic Platforms for Customizable Products, In: Proceedings of the ASME Design Engineering Technical Conferences - Design Automation Conference, Fadel, G., (ed.), Montreal, Quebec, Canada, ASME, Paper No. DETC2002/DAC-34095.
- Hirtz, J., Stone, R, McAdams, D., Szykman, S. and Wood, K., 2002, A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts, Research in Engineering Design, **13**, pp. 65–82.
- Höltkä, K., Tang, V. and Seering, W.P., 2003, Modularizing Product Architectures Using Dendrograms, In: Proceedings of the International Conference on Engineering Design (ICED03), Stockholm, August.
- Huang, C.C. and Kusiak, A., 1998, Modularity in Design of Products and Systems, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems & Humans, **28**(1), pp. 66–77.
- Hundal, M.S., 1997, Systematic Mechanical Designing: a Cost and Management Perspective, New York: ASME Press.
- Iwasaki, Y. and Chandrasekaran, B., 1992, Design Verification Through Function- and

- Behavior-Oriented Representation: Bridging the Gap Between Function and Behavior, In: Proceedings of the Conference on Artificial Intelligence in Design '92, Gero, J.S., (ed.), Kluwer Academic Publishers, pp. 597–616.
- Iwasaki, Y., Vescovi, M, Fikes, R. and Chandrasekaran, B., 1995, Causal Functional Representation Language with Behavior-Based Semantics, *Applied Artificial Intelligence*, **9**, pp. 5–31.
- Jiao, J. and Tseng, M.M., 1998, Fuzzy Ranking for Conceptual Evaluation in Configuration Design for Mass Customization, *Concurrent Engineering: Research and Applications*, **6**(3), pp. 189–206.
- Jiao, J. and Tseng, M.M., 1999, An Information Modeling Framework for Product Families to Support Mass Customization Production, *Annals of the CIRP*, **48**(1), pp. 93–98.
- Jiao, J. and Tseng, M.M., 2004, Customizability Analysis in Design for Mass Customization, *Computer-Aided Design*, **36**, pp. 745–757.
- Jiao, J. and Zhang, Y., 2005, Product Portfolio Identification Based on Association Rule Mining, *Computer-Aided Design*, **37**, pp. 149–172.
- Kahn, H., Filer, N., Williams, A. and Whitaker, N., 2001, A Generic Framework for Transforming EXPRESS Information Models, *Computer-Aided Design*, **33**, pp. 501–510.
- Kimura, F., Hata, T. and Suzuki, H., 1998, Product Quality Evaluation Based on Behaviour Simulation of Used Products, *Annals of the CIRP*, **47**(1), pp. 119–122.
- Kimura, F. and Nielsen, J., 2005, A Design Method for Product Family under Manufacturing Resource Constraints, *Annals of the CIRP*, **54**(1), pp. 139–142.
- Kirschman, C.F. and Fadel, G.M., 1998, Classifying Functions for Mechanical Design,

- 
- ASME Journal of Mechanical Design, **120**(3), pp. 475–482.
- Kohonen, T., 1989, *Self-Organization and Associative Memory*, 3rd ed., New York: Springer-Verlag.
- Li, Z., Liu, M. and Ramani, K., 2004, Review of Product Information Retrieval: Representation and Indexing, In: *Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences*, Paper No. DETC2004/DAC-57749, Salt Lake City, September.
- Liang, W.Y. and Huang, C.C., 2002, The Agent-based Collaboration Information System of Product Development, *International Journal of Information Management*, **22**(3), pp. 211–224.
- Mangun, D. and Thurston, D.L., 2002, Incorporating Component Reuse, Remanufacture, and Recycle into Product Portfolio Design, *IEEE Transactions on Engineering Management*, **49**(4), pp. 479–490.
- Martin, M.V. and Ishii, K., 1996, Design for Variety: a Methodology for Understanding the Costs of Product Proliferation, In: *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, Paper No. 96-DETC/DTM-1610, Irvine, CA., September.
- Martin, M.V. and Ishii, K., 1997, Design for Variety: Development of Complexity Indices and Design Charts, In: *Proceedings of the 1997 ASME Design Engineering Technical Conferences*, Paper No. DETC97/DFM-4359, Sacramento, CA., September.
- Martin, M. and Ishii, K., 2002, Design for Variety: Developing Standardized and Modularized Product Platform Architectures, *Research in Engineering Design*, **13**, pp. 213–235.

- McAdams, D.A., Stone, R.B. and Wood, K.L., 1999, Function Interdependence and Product Similarity Based on Customer Needs, *Research in Engineering Design*, **11**, pp. 1–19.
- McKay, A., Erens, F. and Bloor, M.S., 1996, Relating Product Definition and Product Variety, *Research in Engineering Design*, **8**(2), pp. 63–80.
- Meyer, M.H. and Lehnerd, A.P., 1997, *The Power of Product Platforms- Building Value and Cost Leadership*, New York: The Free Press.
- Michaels, J.V. and Wood, W.P., 1989, *Design to Cost*, New York: John Wiley & Sons.
- Mittal, S. and Frayman, F., 1989, Towards a Generic Model of Configuration Tasks. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Detroit, Michigan, August 20-25, pp. 1395–1401.
- Nayak, R.U., Chen, W. and Simpson, T.W., 2002, A Variation-Based Method for Product Family Design, *Engineering Optimization*, **34**(1), pp. 65–81.
- Nelson, S.A., II, Parkinson, M.B. and Papalambros, P.Y., 2001, Multicriteria Optimization in Product Platform Design, *ASME Journal of Mechanical Design*, **123**(2), pp. 199–204.
- Ong, S.K., Lin, Q. and Nee, A.Y.C., 2006, Web-Based Configuration Design System for Product Customization, *International Journal of Production Research*, **44**(2), pp. 351–382.
- Ostwald, P.F. and McLaren, T.S., 2004, *Cost Analysis and Estimating for Engineering and Management*, Upper Saddle River, NJ: Pearson/Prentice Hall.
- Otto, K. and Wood, K.L., 2001, *Product Design Techniques in Reverse Engineering and New Product Development*, NJ: Prentice Hall.



- 
- Pahl, G. and Beitz, W., 1996, *Engineering Design: a Systematic Approach*, 2nd ed., London: Springer.
- Park, J. and Simpson, T.W., 2005, Development of a Production Cost Estimation Framework to Support Product Family Design, *International Journal of Production Research*, **43**(4), pp. 731–772.
- Pine, B.J., 1993, *Mass Customization: the New Frontier in Business Competition*, Boston: Harvard Business School Press.
- Pratt, M.J. and Anderson, B.D., 2001, A Shape Modeling Applications Programming Interface for the STEP Standard, *Computer-Aided Design*, **33**, pp. 531–543.
- Pugh, S., 1991, *Total Design, Integrated Methods for Successful Product Engineering*, Addison-Wesley Publishing Company.
- Pulm, U. and Lindemann, U., 2001, Enhanced Systematics for Functional Product Structuring, *Design Research- Theories, Methodologies and Product Modelling*, Culley, S. Duffy, A., McMahon, C. and Wallace, K., (eds.), Professional Engineering Publishing, pp. 477–484.
- Qian, L. and Gero, J.S., 1996, Function-Behavior-Structure Paths and Their Role in Analogy-Based Design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**(4), pp. 289–312.
- Rai, R. and Allada, V., 2003, Modular Product Family Design: Agent-based Pareto-optimization and Quality Loss Function-based Post-optimal Analysis, *International Journal of Production Research*, **41**(17), pp. 4075–4098.
- Rezayat, M., 2000, Knowledge-Based Product Development Using XML and KCs, *Computer-Aided Design*, **32**(5), pp. 299–309.
- Roy, U., Pramanik, N., Sudarsan, R., Sriram, R.D. and Lyons, K.W., 2001,

- Function-to-Form Mapping: Model, Representation and Applications in Design Synthesis, *Computer-Aided Design*, **33**, pp. 699–719.
- Sand, J.C., Gu, P. and Watson, G., 2002, HOME: House of Modular Enhancement - A Tool for Modular Product Redesign, *Concurrent Engineering: Research and Applications*, **10**(2), pp. 153–164.
- Sabin, D. and Weigel, R., 1998, Product Configuration Frameworks- A Survey, *IEEE Intelligent Systems and Their Applications*, **13**(4), pp. 42–49.
- Shen, W., Norrie, D.H., and Barthès, J.P., 2001, *Multi-agent Systems for Concurrent Intelligent Design and Manufacturing*, New York: Taylor & Francis.
- Siddall, J.N., 1983, *Probabilistic Engineering Design: Principles and Applications*, New York: M. Dekker.
- Siddique, Z. and Rosen D.W., 2001, On Combinatorial Design Spaces for the Configuration Design of Product Families, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**(2), pp. 91–108.
- Simpson, T.W., 1998, *A Concept Exploration Method for Product Family Design*, Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA.
- Simpson, T.W., 2004, Product Platform Design and Customization: Status and Promise, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **18**, pp. 3–20.
- Simpson, T.W., Chen, W, Allen, J.K. and Mistree, F., 1996, Conceptual Design of a Family of Products through the Use of the Robust Concept Exploration Method, In: *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Published by AIAA, Inc., **2**(2), pp. 1535–1545.

- Simpson, T.W., Chen, W., Allen, J.K. and Mistree, F., 1997, Designing Ranged Sets of Top-Level Design Specifications for a Family of Aircraft: An Application of Design Capability Indices, In: Proceedings of the SAE World Aviation Congress and Exposition, Paper No. AIAA-97-5513, Anaheim, CA., October.
- Simpson, T.W., Rosen, D, Allen, J.K. and Mistree, F., 1998, Metrics for Assessing Design Freedom and Information Certainty in the Early Stages of Design, *ASME Journal of Mechanical Design*, **120**(4), pp. 628–635.
- Simpson, T. W., Maier, J.R.A. and Mistree, F., 2001, Product Platform Design: Method and Application, *Research in Engineering Design*, **13**(1), pp. 2–22.
- Smith, J.S., 2002, A Multiple Viewpoint Modular Design Methodology, Ph.D. Dissertation, Department of Design, Manufacturing and Engineering Management, University of Strathclyde, UK.
- Stone, R.B., Wood, K.L. and Crawford, R.H., 2000a, Using Quantitative Functional Models to Develop Product Architectures, *Design Studies*, **21**(3), pp. 239–260.
- Stone, R.B., Wood, K.L. and Crawford, R.H., 2000b, A Heuristic Method for Identifying Modules for Product Architectures, *Design Studies*, **21**(1), pp. 5–31.
- Stone, R. and Wood, K.L., 2000, Development of a Functional Basis for Design, *Journal of Mechanical Design*, **122**, pp. 359–370.
- Suh, N.P., 2001, *Axiomatic Design- Advances and Applications*, New York: Oxford University Press.
- Suh, N.P., 2005, *Complexity: Theory and Applications*, New York: Oxford University Press.
- Szykman, S., Sriram, R.D. Bochenek, C. and Racz, J., 1998, The NIST Design Repository Project, *Advances in Soft Computing – Engineering Design and*

- Manufacturing, Roy, R., Furuhashi, T. and Chawdhry, P.K., (eds.), London: Springer-Verlag, pp. 5–19.
- Szykman, S., Racz, J.W. and Sriram, R.D., 1999, The Representation of Function in Computer-Based Design, In: Proceedings of the 1999 ASME Design Engineering Technical Conferences, Paper No. DETC99/DFM-8742, Las Vegas, Nevada, September.
- Szykman, S., Fenves, S.J., Keirouz, W. and Shooter, S.B., 2001a, A Foundation for Interoperability in Next Generation Product Development Systems, *Computer-Aided Design*, **33**, pp. 545–559.
- Szykman, S., Sriram, R.D. and Regli, W.C., 2001b, The Role of Knowledge in Next-generation Product Development System, *Journal of Computing and Information Science in Engineering*, **1**, pp. 3–11.
- Ullman, D.G., 1997, *The Mechanical Design Process*, New York: McGraw-Hill.
- Ulrich, K., 1995, The Role of Product Architecture in the Manufacturing Firm, *Research Policy*, **24**, pp. 419–440.
- Ulrich, K. and Eppinger, S., 2004, *Product Design and Development* 3rd ed., Boston: McGraw-Hill/Irwin.
- Ulrich, K. and Seering, W., 1987, Conceptual Design: Synthesis of Systems of Components, In: Proceedings of the 1987 ASME Winter Annual Meeting Symposium on Integrated and Intelligent Manufacturing, Boston, 1987, pp. 57–66.
- Umeda, Y., Takeda, H., Tomiyama, T. and Yoshikawa, H., 1990, Function, Behavior and Structure. In: Proceedings of the AIENG'90 Applications of AI in Engineering, Boston, July, pp. 177–193.
- Watson, I., 1999, Case-Based Reasoning is a Methodology not a Technology.

- Knowledge-Based System, **12**, pp. 303–308.
- Wielinga, B. and Schreiber, G., 1997, Configuration-Design Problem Solving, *IEEE Intelligent Systems*, **12**(2), pp. 49–56.
- William, H.W. and Agogino, A.M., 1996, Case-Based Conceptual Design Information Server for Concurrent Engineering, *Computer-Aided Design*, **28**(5), pp. 361–369.
- Wood, W.H. and Agogino, A.M., 2004, Decision-Based Conceptual Design, Modeling and Navigating Heterogeneous Design Space, *ASME Journal of Mechanical Design*, **127**(1), pp. 2–11.
- Yu, T., Yassine, A. and Goldberg, D., 2003, A Genetic Algorithm for Developing Modular Product Architectures, In: *Proceedings of ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conferences*, Paper No. DETC2003/DTM-48657, Chicago, Illinois, September.
- Yu, J.S., Gonzalez-Zugasti, J.P. and Otto, K.N., 1999, Product Architecture Definition Based upon Customer Demands, *ASME Journal of Mechanical Design*, **121**(3), pp. 329–335.
- Yu, B. and MacCallum, K., 1995, Modelling of Product Configuration Design and Management by Using Product Structure Knowledge. In: *Knowledge Intensive CAD*, Volume 1, Tomiyama (ed.), Springer, pp. 115–124.
- Zamirowski, E.J. and Otto, K.N., 1999 Identifying Product Family Architecture Modularity Using Function and Variety Heuristics, In: *Proceedings of the ASME Design Engineering Technical Conferences*, Paper No. DETC99/DTM-8760, Las Vegas, Nevada.

# APPENDICES

## APPENDIX A FLOW TAXONOMY

Category	Domain	Flow
Energy		friction(01), gravitation(02), centrifugal force(03), contact(04), inertia(05), momentum(06), torque(07), human force(08)
	Mechanical [11]	rotary motion(51), angular displacement(52), angular velocity(53), angular acceleration(54); translation motion(60), position(61), displacement(62), velocity(63), acceleration(64), translation(65); oscillatory(70), combinational(80)
	Electrical [12]	charge(01), wattage(02), electromotive force(03), current(04), voltage impulse(05), electrical impedance (06), resistance(07), capacitance(08), inductance(09)
	Thermal/ Chemical [13]	entropy(01), temperature(02), entropy flow(03), heat(04) combustion(05), oxidation(06), combustible gas-(1307)
	Hydraulic [14]	pressure(01), flow(02), volume(03)
	Optical [15]	reflection(01), refraction(02), diffraction(03), interference(04), polarization(05), infra-red(06), visible(07), ultra violet(08)
Material	Solid [20]	rigid body(01), elastic body(02), widget(03), powder(04), particulate(05), granular-matter(06), composite material(07), aggregate material(08)
	Liquid [21]	incompressible liquid(01), water(02), compressible liquid(03), homogeneous-liquid(04), petrol (05), diesel (06)
	Gas [22]	homogeneous(01), inhomogeneous(02), air(03), oxygen(04) nitrogen(05), carbon dioxide/CO <sub>2</sub> (06), compressible(07), incompressible (08), flammable gas(2209)
	Multi-phase mixture [23]	solid-liquid(01), liquid-gas(02), liquid-particle(03)
Signal	Single [30]	sine wave(01), unit step(02), sinusoid(03), impulse (04), electric signal (05), switch on (06)
	Status [31]	sound(01), temperature(02), pressure(03), verbal(04), tone(05), visual(06), position(07), displacement(08), smell(09)

**APPENDIX B FUNCTION TAXONOMY**

Category		Function Action
Energy and Material	Usage [1]	absorb(01), consume(02), destroy(03), dissipate(04), eliminate(05), empty(06), export(07), remove(08)
		add(21), create(22), emit(23), supply(24), extract(25), generate(26), import(27), provide(28)
		accumulate(41), collect(42), store(43)
	Combination/ Distribution [2]	combine(01), connect(02), couple(03), link(04), mix(05), branch(10), distribute(11), divide(12), separate(13), sort(14)
	Transformation [3]	attenuate(01), convert(02), filter(03), modify(04), refine(05), amplify(11), increase(12), decrease(21)
Conveyance [4]	advance(01), channel(02), conduct(03), convey(04), direct(05), divert(06), guide(07), move(08), rotate(09), transfer(10), translate(11), transmit(12), transport(13)	
Signal	Generation [5]	generate(01), open(02), turn-on(03), emit(04), store-value(05), display(06)
	Processing [6]	adjust(01), decrease(02), delay(03), detect(04), display(05), equalize(06), enhance(07), increase(08), inhibit(09), limit(10), maintain(11), measure(12), resist(13), select(14), sense(15), amplify(16), demodulate(17), attenuate(18), compare(19), decode(20), decrypt(21), digitize(22), encode(23), filter(24), interrupt(25), modulate(26), reset(27), split(28), switch(29), toggle(30), track(31), vary(32), encrypt(33), isolate(34), time(35),
	Logical/ Mathematical [7]	AND(01), NOT(02), OR(03), XOR(04)
		add(11), decrement(12), differentiate(13), divide(14), increment(15), integrate(16), invert(17), multiply(18), shift(19), sort(20), subtract(21)
Elimination [8]	turn-off(01), filtrate(02), close(03)	
Enclosure [9]	assemble(01), constrain(02), cover(03), disassemble(04), enclose(05), extract(06), fasten(07), fix(08), guide(09), join(10), link(11), locate(12), orient(13), position(14), release(15), remove(16), secure(17), separate(18), stabilize(19), support(20), unfasten(21)	