Quality function deployment opportunities in product model supported design

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

Additional Information:

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

Metadata Record: [https://dspace.lboro.ac.uk/2134/13879](https://dspace.lboro.ac.uk/2134/13879)

Publisher: © Abdul Rahman Omar

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

![Creative Commons License](https://creativecommons.org/licenses/by-nc-nd/2.5/)

**Attribution-NonCommercial-NoDerivs 2.5**

You are free:
- to copy, distribute, display, and perform the work

Under the following conditions:
- **Attribution.** You must attribute the work in the manner specified by the author or licensor.
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

For any reuse or distribution, you must make clear to others the license terms of this work.

Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

For the full text of this licence, please go to: [http://creativecommons.org/licenses/by-nc-nd/2.5/](http://creativecommons.org/licenses/by-nc-nd/2.5/)
QUALITY FUNCTION DEPLOYMENT OPPORTUNITIES IN PRODUCT MODEL SUPPORTED DESIGN

By

ABDUL RAHMAN OMAR

A Doctoral Thesis
Submitted in partial fulfilment of the requirements
for the award of

Doctor of Philosophy

of the Loughborough University
Department of Manufacturing Engineering

November 1997

© By ABDUL RAHMAN OMAR, 1997
ACKNOWLEDGEMENTS

I wish to acknowledge and express my sincere thanks to my supervisor, Dr. Keith Popplewell for his supervision, encouragement, suggestions and help throughout this research. His prompt comments and valuable recommendations have helped in the preparation of this thesis. I would like to extend my thanks and gratitude to my director of research Prof. Neil Burns for his continuous support and help.

I would like to express my sincere thanks to Dr. Jenny Harding for sharing her vast experience and knowledge that has made my prototype software workable. Also to all my colleagues of the laboratory GG101, who deserve my thanks for their help and cooperation. My appreciation to those who have directly or indirectly contributed in the preparation of this thesis.

Thanks to Mara Institute of Technology for sponsoring my Ph. D research.

Finally, I wish to express my love and appreciation to my wife, Norlida Mohd. Salleh and my kids: Muhammad Syafiq, Muhammad Sofwan and Muhammad Salman for their constant love, devotion and patience which has taken many of the hours that should have been dedicated exclusively to them. Also, to my late father and my mother for their encouragement and love.
ABSTRACT

QUALITY FUNCTION DEPLOYMENT OPPORTUNITIES IN PRODUCT MODEL SUPPORTED DESIGN

This thesis describes the development of a QFD information model established in an environment where design information is shared between software applications. The main objectives of the research are to establish a QFD information structure within a Product Data Model and to demonstrate how this enables an intelligent, knowledge-based analysis of QFD information contained in a Product Model.

The generic structure of the QFD information has been defined and implemented in prototype software and its value is demonstrated through experimentation in two case studies. Successful implementation of the case studies proved that the QFD information structure is able to capture QFD information as persistent objects residing in a Product Model. It also demonstrates that an intelligent knowledge-based QFD expert can be implemented alongside the QFD information model to accomplish useful, consistent, reasoned analysis of QFD information.

The research has achieved its aim to provide a new contribution in the product design domain, and to the effectiveness of Concurrent Engineering activities, through better use of Quality Function Deployment.

Keywords: Quality Function Deployment, QFD, Information Modelling, Knowledge Representation Model, KRM, Concurrent Engineering, CE.
ABBREVIATIONS

AI  Artificial Intelligent
ASI  American Supplier Institute
CACE  Computer-Aided Concurrent Engineering
CE  Concurrent Engineering
DARPA  Defence Advanced Research Projects Agency
DBMS  Database Management System
ddl  Data definition language
EM  Engineering Moderator
GUI  Graphical User Interface
HoQ  House of Quality
KRM  Knowledge Representation Model
MOSES  Model Oriented Simultaneous Engineering System
OO  Object Oriented
QFD  Quality Function Deployment
QFD DSS  Quality Function Deployment Decision Support System
SE  Simultaneous Engineering
SQL  Structure Query Language
VCE  Virtual Concurrent Engineering
VOC  Voice of Customer
LIST OF FIGURES

Figure 1-1 The Structure of the Thesis 9
Figure 2-1 The traditional “Over The Wall Approach” to product innovation [Parrot, 1995] 13
Figure 2-2 Sequential Method Versus Concurrent Engineering 15
Figure 2-3 Evolution of Software [Pressman, 1994] 22
Figure 2-4 The Four Phases of QFD Process [Lo et al, 1991] 29
Figure 2-5 The House of Quality (HoQ) [Hauser et al, 1988] 30
Figure 3-1 Overall Research Methodology 43
Figure 3-2 MOSES Computational Viewpoint 47
Figure 3-3 The MOSES Architecture and Integrated QFD 48
Figure 3-4 Using the QFD Information Model 48
Figure 3-5 The class structure of NEEDS_aro 49
Figure 3-6 Objectivity/DB Logical Storage Model 51
Figure 3-7 Research Focus and its Environment 53
Figure 4-1 Class Diagram of the QFD Information Structure 58
Figure 4-2 Type Hierarchy and Inheritance for Class CUSTOMER_NEEDS and Sample Rule Set for Frame Generation and Assignment of Slot Values [Bird, 1992] 64
Figure 4-3 QFD Matrix for a fuel Pump [Sriraman et al, 1990] 65
Figure 4-4 The main Menu Window of the HoQ 67
Figure 4-5 The Selection Menu Window 67
Figure 4-6 A Typical Objectivity/DB Tool Manager Browser 68
Figure 4-7 A Typical Example of Attributes Listed in Basic Object 69
Figure 4-8 A Simple Flow Diagram of a QFD Process 74
Figure 4-9 The Selection Box for Product Object Name 75
Figure 4-10 The Selection Box for Creating Secondary Needs Option 76
Figure 4-11 The Selection Box for Primary Needs Object 76
Figure 5-1 The Knowledge Representation Model [Popplewell et al, 1995] 82
Figure 5-2 KRM Software Expert Structure [Harding, 1996b] 82
LIST OF FIGURES

Figure 5-3    A Representation of Knowledge, Captured within Rule and RuleSet Classes, Using Booch Object Oriented Design Graphical Notation [Harding, 1996b] 83
Figure 5-4    The Class Structure of the QFD Software Expert Using Booch Object Oriented Graphical Notation 84
Figure 5-5    A Representation of the Expression Class, of the QFD Software Expert, Using Booch Object Oriented Notation 85
Figure 5-6    A Representation of Simple_Action Class of QFD Expert Software, Using Booch Object Oriented Graphical Notation 86
Figure 5-7(a) The Selection Window for Rules Creation 88
Figure 5-7(a) The Selection Window for Rule Type 88
Figure 5-8    The Selection Window for Display Mode 89
Figure 5-9    The Text Input Window 89
Figure 5-10(a) The Selection Window for Resulting Action Option 89
Figure 5-10(a) The Selection Window for Knowledge Rule Option 89
Figure 5-11   The Implementation of Rule Class for the Expert 90
Figure 5-12   Knowledge Base Object 91
Figure 5-13   HOQ_Working_Memory Object 92
Figure 6-1 (a) A Portion of the House of Quality Showing the Captured Information About a Mug 105
Figure 6-1 (b) The Other Half of the House of Quality Showing the Captured Information About a Mug 106
Figure 6-2    The Database Browser Showing an Implementation of Sub-Classes of Simple Action Class to Determine Design Features’ Priorities 109
Figure 6-3 (a) The Output for Initial Coefficient Values After Applying KRM’s First Rule 110
Figure 6-3 (b) The Output of Coefficient Values After Applying KRM’s Second Rule 111
Figure 6-3 (c) The Output of Coefficient Values After Applying KRM’s Third
LIST OF FIGURES

Rule  112
Figure 6-3 (d) The Output of Coefficient Values After Applying KRM's Fourth Rule  113
Figure 6-3 (e) The Output of Coefficient Values After Applying KRM's Fifth Rule  114
Figure 6-3 (f) The Output of Coefficient Values Listed in Order of Priority  115
Figure 6-4 A Display of a Section of Specification Correlation Matrix for Mid-Range Hi-Fi Equipment  122
Figure 6-5 A Display of Primary Needs, Secondary Needs and a Section of the Interrelationship Matrix for Mid-Range Hi-Fi Equipment  123
Figure 6-6 A Display of Customers' Needs Benchmark for Mid-Range Hi-Fi Equipment  124
Figure 6-7 A Display of Specification Priority Listing for a Mid-Range Hi-Fi Equipment  125
Figure 7-1 The Initial QFD Information Model Structure  128
Figure 7-2 The PRIMARY_NEEDS Class and its Associates Added to the Initial QFD Information Structure  130
Figure 7-3(a) Initial Implementation Part of the Coding for PRODUCT Class in ddl File  131
Figure 7-3(b) Modified Coding for PRODUCT Class in ddl File  132
Figure 7-4 The Display Output of the Focus HoQ of a Mug  134
Figure 7-5(a) The Output Display of Initial Coefficient Values for Aspect Priority Analysis of a Mug  135
Figure 7-5(b) The Output Display of Coefficient Values After Applying the Second KRM's Rule for Aspect Priority Analysis of a Mug  135
Figure 7-5(c) The Output Display of Coefficient Values After Applying the Third KRM's Rule for Aspect Priority Analysis of a Mug  135
Figure 7-5(d) The Output Display of Coefficient Values After Applying the Fourth KRM's Rule for Aspect Priority Analysis of a Mug  136
LIST OF FIGURES

Figure 7-5(e)  The Output Display of Coefficient Values After Applying the Fifth KRM’s Rule for Aspect Priority Analysis of a Mug  136
Figure 7-5(f)  The Output Display of Coefficient Values Listed in Order of Priority  136
Figure 7-6  The Output Display of the Consolidated HoQ of a Mid-Range Hi-Fi Equipment  138
Figure 7-7(a)  The Output Display of the Consolidated HoQ Coefficient Values for Mid-Range Hi-Fi Equipment  139
Figure 7-7(b)  The Output Display of the Consolidated HoQ Coefficient Values of Mid-Range Hi-Fi Equipment Listed in Order of Priority  139
### LIST OF TABLES

| Table 1-1 | The Research’s Relation to Research done by Harding and Fung | 6 |
| Table 4-1 | An Extract of the House of Quality for a Mug | 59 |
| Table 4-2 | Elements of QFD Information Model Class Structure for PRODUCT, PRIMARY_NEEDS, NEEDS, ASPECT and SPECIFICATION Class | 59 |
| Table 4-3 | Elements of QFD Information Model Class Structure for Interrelationship and Correlation Matrices | 61 |
| Table 4-4 | Elements of QFD Information Model Class Structure for Competitor and its Benchmarks | 62 |
| Table 4-5 | Weightage Scale for Customer Needs Importance [Ginder, 1991] | 72 |
| Table 4-6(a) | Numerical Weightage for Relationship Matrix | 72 |
| Table 4-6(b) | Numerical Weightage for Correlation Matrix | 72 |
| Table 4-7 | Benchmark Scale for Competitive Assessment [Ginder, 1991] | 72 |
| Table 4-8 | The Difficulty Rating Scale [Shillito, 1994] | 72 |
| Table 5-1 | Knowledge that Checks and Updates Each Coefficient Values if Need Score for Own Product is Greater Than Competitor Need Score | 93 |
| Table 5-2 | Knowledge that Checks and Updates Each Coefficient Values if Specification Score for Own Product is Greater Than Competitor Specification Score | 94 |
| Table 6-1 | List of Customer Needs, Customer Relative Importance and Competitors’ Benchmark | 98 |
| Table 6-2 | List of Aspects and Specifications that Contemplate Customer Needs for a Mug Design | 101 |
| Table 6-3 | The Technical Competitive Benchmark Values | 102 |
| Table 6-4 | Correlate Design Features to Customer Secondary Needs | 103 |
| Table 6-5 | Mug Design - Identify Correlation Matrix | 104 |
| Table 6-6 | Rules to Determine Design Attribute’s Priority | 108 |
| Table 6-7 | The Information of Customers’ Needs and Competitive Benchmark | 104 |
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table 6-8</th>
<th>The Information of Design Features' and Competitive Technical Benchmark Values for a Mid-Range Hi-Fi Equipment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Fung, 1997]</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Certification of Originality ........................................... i  
Acknowledgements ....................................................... ii  
Abstract ..................................................................... iii  
Abbreviations ................................................................ iv  
List of Figures ................................................................ v  
List of Tables ................................................................ ix  

## CHAPTER ONE: INTRODUCTION

1.1 RESEARCH BACKGROUND AND PROBLEMS .................. 1  
1.2 RESEARCH REQUIREMENTS ..................................... 4  
1.2.1 Research Link .................................................. 5  
1.3 POTENTIAL BENEFITS FROM THE RESEARCH STUDY ... 6  
1.4 ORGANISATION OF THE THESIS ............................... 8  

## CHAPTER TWO: LITERATURE SURVEY

2.1 INTRODUCTION ......................................................... 11  
2.2 TRENDS AND FUTURE REQUIREMENTS IN CE SYSTEMS 11  
2.2.1 Evolution of Concurrent Engineering at a Glance ... 11  
2.2.2 Factors Affecting the Changes ............................... 18  
2.3 CURRENT APPROACHES TO SOFTWARE DESIGN ON CE SYSTEMS 22  
2.3.1 Current Approaches in Software Design .................. 22  
2.3.2 Present Software Design of Applications for CE Systems 23  
2.4 POTENTIAL AREAS FOR FUTURE ENHANCEMENT OF CURRENT SOFTWARE DESIGN IN CE ............ 25
# TABLE OF CONTENTS

2.5  OPPORTUNITY FOR INCLUDING QUALITY FUNCTION DEPLOYMENT (QFD) IN CURRENT SOFTWARE DESIGN FOR CE SYSTEMS  
2.5.1 QFD Theoretical Concept  
2.5.2 Benefits of QFD  
2.5.3 Potential Reinforcement of Current QFD Implementation  

2.6  SUMMARY  

CHAPTER THREE: RESEARCH METHODOLOGY AND PROCEDURES  

3.1  INTRODUCTION  

3.2  ISSUES INVOLVED IN DESIGNING A QFD INFORMATION MODEL  
3.2.1 Modelling QFD System  
3.2.2 Database Management System  
3.2.3 Integratability  
3.2.4 Graphical User Interface (GUI)  

3.3  STRUCTURING RESEARCH REQUIREMENT  
3.3.1 Analysis and Selection of Architecture  
3.3.2 Selection of Customers’ Needs and Design Features  

3.4  CONSTRUCTION OF THE DECISION SUPPORT SYSTEM FOR THE QFD INFORMATION MODEL
TABLE OF CONTENTS

3.5 EXPERIMENTAL ENVIRONMENT 47

3.5.1 The Model Oriented Simultaneous Engineering System (MOSES) Architecture 47
3.5.2 Programming Language 49
3.5.3 Database System 51
3.5.4 Motif Environment 52

3.6 RESEARCH FOCUS 52

3.7 SUMMARY 54

CHAPTER FOUR : DEVELOPMENT OF A QUALITY FUNCTION DEPLOYMENT INFORMATION MODEL 55

4.1 INTRODUCTION 55

4.2 SELECTION OF QUALITY FUNCTION DEPLOYMENT (QFD) INFORMATION MODEL ARCHITECTURE 56

4.2.1 Background 56
4.2.2 QFD Information Model Structure 56
4.2.3 Implementation of Data Structure 62

4.3 APPLICATION AND EVALUATION OF THE QFD INFORMATION MODEL 63

4.3.1 Characteristics of the QFD Information Model 63
4.3.2 QFD Information Model Assessment 64
# TABLE OF CONTENTS

4.4 SPECIFICATION OF THE QFD INFORMATION IN PRODUCT MODEL  
4.4.1 Graphical User Interface  
4.4.2 Getting the Correct Facts into HoQ  
4.4.3 Procedures to Capture QFD Information

4.5 SUMMARY

CHAPTER FIVE: DEVELOPMENT OF QFD DECISION SUPPORT SYSTEM

5.1 INTRODUCTION

5.2 DEVELOPMENT AND REALISATION OF KNOWLEDGE BASED REPRESENTATION MODEL  
5.2.1 Background  
5.2.2 The Knowledge Representation Model Concept

5.3 SELECTION OF THE QFD EXPERT SYSTEM ARCHITECTURE AND ITS METHODS  
5.3.1 QFD Software Expert Architecture  
5.3.2 Working Memory Class for QFD Software Expert  
5.3.3 Knowledge Base Class for QFD Software Expert  
5.3.4 Implementation of Data Structure for QFD Software Expert
# TABLE OF CONTENTS

5.4 THE IMPLEMENTATION OF THE KNOWLEDGE REPRESENTATION MODEL  

5.4.1 Populating the Knowledge Representation Model 87  

5.4.2 Algorithm to Determine Mean Value for Interrelationship Weighting Scale 90  

5.4.3 Algorithm to Determine Design Feature Priority for QFD Software Expert 92  

5.4 SUMMARY 95  

CHAPTER SIX : ASSESSMENT OF QFD INFORMATION AND SOFTWARE EXPERT MODELS 96  

6.1 INTRODUCTION 96  

6.2 CASE STUDY 1 : A MUG DESIGN 96  

6.2.1 Customer Needs, Relative Importance and Competitive Evaluation 97  

6.2.2 Development the Technical Portion of the HoQ 99  

6.2.3 Analysis of the HoQ and Priorising Design Attributes 107  

6.3 CASE STUDY 2: A MID-RANGE HI-FI SYSTEM DESIGN 116  

6.3.1 Populating the Information in the Product Model Database 116  

6.3.2 The Output Display of the Hi-Fi Equipment Populated in the Product Model 121
# TABLE OF CONTENTS

6.3.3 Analysis of the HoQ to Prioritise Design Attributes ........................................... 121

6.4 SUMMARY ...................................................................................................................... 126

CHAPTER SEVEN: ENHANCEMENT OF THE QFD INFORMATION MODEL AND ITS POTENTIAL BENEFITS ............................ 127

7.1 INTRODUCTION ............................................................................................................. 127

7.2 QFD INFORMATION MODEL - AN INTEGRATED OBJECT ORIENTED PRODUCT MODEL ............................................. 128

7.2.1 Initial QFD Information Model .................................................................................. 128

7.2.2 The Implementation of the Modify QFD Information Structure ............................... 129

7.2.3 Future Expansion of the Software ............................................................................. 132

7.3 ENHANCEMENT OF THE CAPTURED QFD INFORMATION IN THE PRODUCT MODEL .............................................. 132

7.3.1 Example 1: QFD Information Enhancement on Mug Design ................................ 133

7.3.2 Example 2: QFD Information Enhancement on Mid-Range Hi-Fi Equipment ........ 137

7.4 BENEFITS FROM USING THE QFD MODEL AND ITS ENHANCED APPLICATION ..................................................... 140

7.4.1 Flexibility Enhancement ............................................................................................ 140

7.4.2 Efficient Development of Software and its Application ........................................... 141

7.5 SUMMARY ...................................................................................................................... 141
# TABLE OF CONTENTS

CHAPTER EIGHT: CONTRIBUTIONS TO KNOWLEDGE AND ISSUES FOR FURTHER INVESTIGATION 143

8.1 INTRODUCTION 143

8.2 CONTRIBUTIONS TO KNOWLEDGE 143

8.2.1 QFD Information Founded Upon a Product Model 143

8.2.2 Knowledge-based QFD Expert Analysis 145

8.2.3 Architecture of QFD Modelling System 146

8.2.4 Integration of the QFD Information Model With Other Information Models 146

8.2.5 Promotion of Product Innovation 147

8.3 RECOMMENDATIONS FOR FUTURE WORK 148

8.3.1 Establishment of Links With Other Concurrent Engineering Systems 148

8.3.2 Enhancement of the Intelligent Knowledge-based System 149

8.3.3 Enhancement of the Generic Intelligent Knowledge-based System 149

8.3.4 Enhancement of the Graphical User Interface Capability 150

8.3.5 The Need for Improved Database System 150

8.3.6 Development Information Structure for the Remaining QFD Phases 151

CHAPTER NINE: SUMMARY AND CONCLUSION 152

9.1 SUMMARY 152

9.2 CONCLUSION 154
# TABLE OF CONTENTS

## REFERENCES 156

Bibliography 168

## APPENDIX I - DATA DEFINITION FILE

- needs_spec.ddl 213
- inter_correlation.ddl 218
- benchmark.ddl 221
- primary_needs.ddl 224
- rules.ddl 228
- working_memory.ddl 237
- mod_rules.ddl 240
- mod1_rules.ddl 246

## APPENDIX II - Lists of Knowledge Representation Model's Rules 249
1 INTRODUCTION

1.1 RESEARCH BACKGROUND AND PROBLEMS

Fast moving changes in business environments have transformed the basis of industrial competitiveness. Today, economic success of a manufacturing enterprise is associated with its ability to produce quality products which satisfy market demands yet incur low development cost and short time to market. In a manufacturing enterprise, many factors can effect levels of performance including: design response time, operating costs, production, manufacturing and distribution of products. The manufacturing enterprise must be able to respond quickly to changes in the market, to new demands or to new opportunities even before the inception of the product in the market.

The introduction of the Concurrent Engineering (CE) concept over recent years has acted as a catalyst for new product development processes and recent enhancements of these concepts, such as the emergence of computer simulation theories in CE, such as virtual concurrent engineering (VCE), have opened up new horizons in this competitive world. An enterprise may need, especially during product development, to assess the use of emerging technology that will enhance productivity and competitiveness. Implementation of this concept, in which software is embedded, is a key requirement for the achievement of an enterprise's competitive goals, as this leads to a reduction in the time-to-market of products. It will also encourage a closer link between team members as virtual co-location will function as an alternative to the ineffective communication
and data exchange which can result from widespread physical location of the different functions of the manufacturing enterprise.

As competition intensifies and technological changes affect both products and processes, standard practice, as offered by the traditional approach, will become less and less effective. The need to break down the wall between design and manufacturing is now eminently necessary. Competitiveness will prevail if the manufacturing enterprise can successfully link and interact with its customers and suppliers. Profits and growth can only be sustained when customer needs are met or exceeded. Success in the marketplace will require reassessment of total business strategy and the tools used to pursue it.

Research has also shown that world class manufacturing organisations are better able than their competitors to align their products and services to customer needs [Adiano et al, 1994]. Enterprises are beginning to realise the importance of responding to identified needs of the market rather than developing products which are forced on to customers [Bennett et al, 1992]. There is a need to explore techniques that can offer options to integrate customer needs into product design development and deliver those products in the most competitive fashion.

Quality Function Deployment (QFD) [Kogure et al, 1983; Sullivan, 1986a; Hauser et al, 1988; Brown, 1991] has been promoted as a tool that ensures customers' needs are satisfied and translated throughout the product design and development process and has set a milestone on the road towards achieving the enterprise's goals in the competitive market. In order to improve product quality in reduced time scales, the research reported here investigates ways of incorporating a QFD software expert into an extended CE design team in order to focus design effort directly on aspects of the product that most effectively enhance customer satisfaction. Amalgamating the concept of VCE and QFD into an integrated environment provides a powerful tool to help introduce more competitive products into the market.
A coherent approach using QFD techniques, capable of fitting design requirements against possible design features during the early stages of the design process, has been shown to save product development time [Sullivan, 1986a]. Company resources are focused on providing capabilities which produce customer satisfaction and hence enhance product quality. At the same time inter and intra department communication is improved which helps to reduce product introduction and development timescales.

In practice, however, existing QFD implementations have limitations that must be addressed before the technique can effectively be used in engineering design. Such remaining problems that have discouraged many potential users are summarised as follows:

(1) As systems become larger, analysis of the data becomes more difficult because of the magnitude of the resulting QFD matrix [Daetz, 1989].

(2) It has become almost impossible to record the QFD matrix manually in a paper form, and in particular to modify the matrix in the light of subsequent change [Wolfe, 1994].

(3) There is a lack of intelligent software tools that can provide useful, consistence, reasoned analysis of QFD information [Syan et al, 1994].

The introduction of computer packages has been proposed as a tool to automate repetitive and tedious operations or processes but their implementation for QFD systems has proved to be successful only for small systems. However for larger and complex systems its effectiveness is still to be seen. Some of the research work reported have started to address these issues seriously. The author believes that the QFD technique will continue to be developed as it matures as a tool for product development and design and as investigations wish to examine more complex systems.
1.2 RESEARCH REQUIREMENTS

In providing solutions to the above problems, it is necessary to establish a representation of the QFD system in the form of information structure to enable the effective application of QFD concepts in product design and development. With the advance of computer technology and associated software tools it is now possible to capture and store knowledge in a consistent structured manner and with acceptable access times. New emerging computer technologies and software tools offer an environment for the development of a QFD information system architecture as part of a comprehensive computer aided engineering architecture to support CE teams. The computer aided engineering architecture promotes information sharing, thus information structures are envisaged as part of a comprehensive Product Model, that supports an effective application of QFD concepts within product design and development.

Hence the primary research questions being investigated in this study are:

1. Is the integration of QFD Support Software with other CE software agents feasible and able to contribute to the process of product innovation?

2. Can the integrated software ease problems associated with the analysis of the QFD information?

This research seeks to use architectural frameworks, which facilitate the structuring of QFD information with Object Oriented (OO) methodology [Booch, 1994] as its main supporting tool. This can provide generalised definitions of the information structure which are modular, integrated, reusable and extendable. The objective is to create a computer software environment which enables the efficient implementation of the QFD information and this will be tested through experimentation with case studies.

A further objective of this work is to show that when the software tool (interfaced to a QFD database), is also interfaced to product design information within a Product...
Model (repository of product, see chapter 2) database, it can become a valuable design aid. This information created and required during the design process can be shared between design team members thus giving CE team members the ability to handle customer requirements and the QFD information model throughout the design process.

1.2.1 Research Link

The proposed QFD information structure is envisaged to provide a software environment that can generate and store QFD information within a Product Model. The value of the captured QFD information will be optimised if users are able to utilise and manipulate the information. This understanding has led the author to explore the possibility of enhancing the QFD information structure to include knowledge-based capabilities, in an integrated environment, to analyse the QFD information.

The author has utilised the Knowledge Representation model developed by Harding [1996b] to capture QFD expert knowledge. The structure of the KRM enables any software expert to be developed and built as an instance of the KRM. Hence the QFD software expert is developed as an instance of the Software Expert class inherited from the KRM. Table 1-1 illustrates the relationship of this research with research established by Harding.

The KRM approach has enabled the expert knowledge of the QFD software expert to be captured using a production rule approach. Here the author used mathematical algorithms as rules populated in the KRM database to determine design features’ priority. However other algorithms such as neural network and fuzzy logic sets can be developed and populated as captured knowledge in the KRM to analyse the QFD information.

For example, Fung (1997) uses Fuzzy Set Theory to develop an algorithm to interpret customer attributes (customer requirements and customer relative important)
into appropriate product attributes using fuzzy inference. This algorithm can be
developed and captured as instance of the KRM to analyse the QFD information. The
author has used Fung’s case study material in his research implementation.

<table>
<thead>
<tr>
<th>Research Work</th>
<th>Focus</th>
<th>Research Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omar</td>
<td>QFD Information</td>
<td>Enable QFD information to be captured and stored in a</td>
</tr>
<tr>
<td></td>
<td>Structure</td>
<td>Product Model, and enables its subsequent expert analysis.</td>
</tr>
<tr>
<td>Harding</td>
<td>KRM</td>
<td>Provides an environment that facilitates the capture of QFD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expert Knowledge to support QFD analysis.</td>
</tr>
<tr>
<td>Fung</td>
<td>Hybrid System for</td>
<td>Uses Fuzzy Set Theory to develop an algorithm to</td>
</tr>
<tr>
<td></td>
<td>Customer Requirement</td>
<td>analyse customer attributes and determine product</td>
</tr>
<tr>
<td></td>
<td>Analysis and Product</td>
<td>attributes' target. Case study material has been</td>
</tr>
<tr>
<td></td>
<td>Attribute Targets</td>
<td>utilised to demonstrate the value of the QFD</td>
</tr>
<tr>
<td></td>
<td>Determination</td>
<td>information structure.</td>
</tr>
</tbody>
</table>

Table 1-1 The Research Relation to Research done by Harding and Fung

1.3 POTENTIAL BENEFITS FROM THE RESEARCH STUDY

The proposed research study is motivated by the development of the state-of-art information technology and computer systems. The technology advancement has encouraged large amounts of research work to focus on studying real systems and virtually simulated them in computer system. Hence, the objective of this research is to
provide a methodology for the development of QFD information structure in a Product Model and to provide an intelligent knowledge-based analysis of QFD information contained in the Product Model. The modelling and simulation environment and tool so conceived are expected to provide:

- an improved means of capturing QFD information during the product development stage, thus potentially saving the high cost of discovery of customer needs after the product has been manufactured;
- an improved integratability and extendability of the information model, through the establishment of formal and flexible methods of capturing information;
- an improved means of analysing QFD information in a useful, consistent, reasoned manner

Furthermore, the research work will propagate new knowledge that extends beyond the existing Product and Manufacturing Models (a Manufacturing Model is a repository of manufacturing activities, see chapter 2) within the CE concept. The concept and research methodology presented will thus be the main contribution to the implementation, integration and moderation of an intelligent software agent in a CE environment. QFD system integration with Product and Manufacturing Models shall provide a new horizon to the CE work done so far. Customers will be able to give feedback and input at the very first crucial stage in the design process. The proposed model will offer a better understanding about effects of customer desire in achieving a competitive and value added product.

Development of QFD expert knowledge as proposed in the research work will provide a conducive platform for manufacturing enterprise to implement QFD system. QFD expert knowledge will assist user in developing a comprehensive QFD system. Managers shall be able to give more attention to engage their expertise solving actual problems rather than being involved with technicalities.

The QFD information system once developed could also be used as a teaching
toolkit in higher learning institutions. The QFD information system will provide a conceptual framework or a model and guidelines for designers and project managers to be aware of the importance of customer requirements being incorporated into the designs. It will serve as a design tool that scans through all the processes involved in the manufacturing enterprise. Hence, better appreciation of the underlying relationships between customers and manufacturing companies could be achieved.

The CE concept employed in integrating QFD system with existing Product and Manufacturing Models will therefore form the main focus of the research work.

1.4 ORGANISATION OF THE THESIS

The thesis is organised into nine chapters including chapter 1. Figure 1-1 illustrates the overall organisation of the thesis in sequence and established the interaction between chapters. Chapter 1 has outlined the general background, research requirements and potential benefits that could be exploited from the research study.

Chapter 2 reviews the literature on the development, utilisation and current approaches to product design. It describes the trend of current research to model and simulate these product design approaches by utilising the potential capabilities of computer technology. Furthermore this chapter investigates the opportunity of including QFD methodology in the process. An understanding about the QFD methodology, its definition, its benefits and its potential enhancement are also established in this chapter.

Chapter 3 explains the research methodology and procedure. It defines issues involved in designing a QFD information model and then describes steps taken to structure the research requirement. Artificial Intelligent (AI) support that is required in order to provide comprehensive analysis of the QFD information is considered in this chapter. This chapter also relates the necessary tools required to accomplish the research objectives described in chapter 1 and discusses the focus of this research.
Figure I-1. The Structure of the Thesis
Chapter 4 demonstrates a novel methodology which formally structures the QFD information within a Product Model. It outlines the selection of a QFD information model architecture and discusses the application and evaluation of the QFD information model with current approaches. Steps and procedures to capture QFD information leading to the process of populating the database are discussed.

The identified QFD information model has made it possible for intelligent software solutions to support the analysis of the QFD information. Hence chapter 5 discusses the development of QFD decision support system that facilitates the analysis of information captured by QFD information model. It utilises Knowledge Representation Model (KRM) architecture to facilitate the design and implementation of hybrid software experts.

In chapter 6, the performance of the QFD information model is evaluated in respect of its capability to support product design. The investigation of the QFD information model through two case studies demonstrates the value and flexibility of the structure. This has been proved further through the analysis of its enhanced capabilities discussed in chapter 7, which considers the analysis of a simplified version of QFD information by utilising first level of customer needs and design features information.

Chapter 8 critically assesses the merits of the research and suggests possible and further research work. Finally chapter 9 summarises key issues described in this thesis pertaining to the development of the QFD information model and concludes the overall research work.
2.1 INTRODUCTION

This chapter considers typical CE application domains in manufacturing enterprises and thereby sets the scene for this research. The literature survey examines various aspects of current CE. Particular reference is made to Quality Function Deployment (QFD) which has emerged as a powerful and effective tool that shows promise in its usage in many engineering applications with high levels of inherent complexity.

2.2 TRENDS AND FUTURE REQUIREMENTS IN CE SYSTEM

2.2.1 Evolution of CE at a Glance

The development trend of the CE can be clearly distinguished as a sequence of phases described as follows:

a. Sequential Engineering

After the Second World War when the industrialised world experienced the growth decades, major manufacturing industries evolved into giant bureaucracies.
Chapter 2

[Backhouse, et al, 1996]. This occurred during the time when industries started to produce consumer products. Backhouse further identified that as there was no competition in the consumer market where the demand began to outstrip supply, companies found they could remain highly profitable without having to work too hard. They divided themselves into specialist functions and began to follow a sequential design process.

However Prasad (1996) has different view on how the sequential concept of manufacturing process was introduced. He said that manufacturing enterprise was led to believe that control of information flow was the key component to success and hence adequate authority was given to functional organisation in the enterprise to manage the flow of information. Product was task-focused and engineering was sequencing-based.

Even though some authors have different views on how sequential based activities in manufacturing industries started, they agree that sequential processes were once adopted by manufacturing industries in the product development process. This traditional manufacturing industry approach to design has been typified by the "over the wall" approach [Parrot, 1995] as illustrated by figure 2-1 where the functions in the manufacturing activities are isolated from one another and they progress in a sequential manner whereby inputs from upstream departments must be completed and pass to the next department before this department can start its process. Furthermore, the strong vertical relationships that exist within individual departments encouraged the development of specialities and cause departments to be isolated physically and organisationally. Departments tended to become autonomous with respect to mainstream design functions [Hang,1990]. Instead of designing and engineering the product in a functional and manufacturable configuration at the very outset of product inception, improvements are made after the product reaches the customers [Dorf et al, 1994].
Prasad (1996) concluded some of the major problems that manufacturing engineers encountered:

- Unsuitable product design for production
- Unavailability of adequate manufacturing equipment
- Tight tolerance which could lead to extra work and high scrap generation
- Problems with parts assembly
- Inability to utilise the existing production equipment, tooling, automatic assembly, etc.

Very often the individual functions in an organisation within the manufacturing enterprise obtained feedback from the manufacturing activity in the form of corrections, changes and errors. These changes were put back to the drawing boards for alterations. Consequently, the product took greater time to market and became more expensive as it required redesigning during the cycle.
b. Concurrent Engineering (CE)

Over the years, as consumer products’ market began to expand, more and more manufacturing enterprises ventured into this profitable business, and competition became inevitable. The sequential approach of traditional manufacturing could no longer assist the enterprise in coping with new development. The increasing competition in the world market place forced companies to look for new means of improving product quality, decreasing product costs and reducing time to market.

The concept of CE was proposed to improve traditional product development practice. CE which is also known as Simultaneous Engineering (SE) is a concept that started to became popular in the late 1980s’. Among those whose pioneered the implementation of this concept was the United States Defence Advanced Research Projects Agency (DARPA) [Syan et al, 1994] who defined CE as:

"Concurrent Engineering is a systematic approach to integrated concurrent design of products and their related processes including manufacturing and support. This approach is intended to cause the developers from the outset to consider all elements of the product life-cycle from conception through disposal including quality, cost, schedule and other user requirements".

The underlying principle of concurrent engineering is to satisfy functionality, reliability, producibility and marketability simultaneously, and is targeted at reducing product development time and cost, achieving higher product quality and value [Hon et al, 1994]. Figure 2-2 shows the management of information flow in CE (depicted in figure 2-2(b)) as compared to the sequential method (depicted in figure 2-2(a)).
A successful implementation of CE program must involve all departments within the enterprise in a team approach, starting at concept design which is the crucial stage in the product life-cycle. It is very critical to define design parameters at this design stage as they are still on paper. Research has shown that up to 70% of a product manufacturing cost is dictated by decisions made during the product design stage [Young, et al, 1992; Syan et al, 1994]. Hence, the key aspect of the CE concept is to integrate all the resources in the enterprise that perform the design activities very early in the design process.

However to support this involvement, and allow the multi-disciplinary teams which comprise of marketing department, design department, production department, manufacturing department, materials department and the management to contribute efficiently and effectively as the design develops, working in close proximity is desirable. Technical data about the product can be exchanged among the multi-
discipline team members and any matters arising as a result of design misalignment can be rectified immediately by these team members.

This type of CE practice could be managed effectively and the target goals could be achieved if the companies are small, where all their operations are done in-house and they have very skilled and experienced people in the organisation to get the job done. However in an increasing global manufacturing environment getting all the team members working in close proximity is not easy and may even be uneconomical [Popplewell et al, 1995]. In the present scenario in the global manufacturing environment it is almost common for a manufacturing enterprise to have their technical expertise distributed across countries or even continents. The task of CE will become more difficult and complex as it becomes necessary to transfer design information to these distributed locations. Hence it is desirable to provide support in various tasks of product development for improving performance.

(e) Virtual Concurrent Engineering

In an attempt to encourage the development of CE, virtual co-location was proposed [Popplewell et al, 1995; Jagannathan et al, 1991]. The disciplines of the product design team can have the ability to interact with one another through a live environment possibly throughout the world.

With recent advances in computing technologies, information processing theories and engineering techniques, it is now possible to simulate the co-location of the design team [Molina et al, 1994a; Morenc et al, 1992; Lindeman et al, 1992]. Manufacturing enterprises can now take advantage of hardware and software support to implement successful CE. Among other factors, networking, electronic data interchange, computer assistance and communication technologies such as e-mail, fax, video-conferencing have stimulated increasing use of advanced systems to support CE.
Each discipline that is part of the CE team can use computer hardware and software tools to analyse and to keep track of product related data. These tools allow easy reconfiguration and extension of a system by offering systematic and modularised knowledge of manufacturing activities and make computer support activities transparent for human understanding [Kimura, 1993]. Each discipline can be housed on different site or even use different computer systems and work independently, only to communicate when information is required from other disciplines to make a decision.

However the multi-platform, multi-site computer systems used by each discipline must be able to communicate with each other to exchange necessary information. Much of the research work currently undertaken [Gallagher, 1991; Hon et al, 1994] has been to integrate an environment that satisfies the requirements of each individual team member so that they may store and maintain information accurately. Hence data standards are an important issue in virtual co-location CE.

Another important issue addressed by the virtual co-location CE concept is that all team members should be able to co-operate with each other, and be aware at which point they should be contributing. They should also be able to know about decisions made by other team members that impinge upon their area of interest. The designers are aware of critical decisions that must be made and ensure that proper communication takes place. However, it will become increasingly difficult to manage the information as the projects become more complex and the need to implement intelligent software support to the CE project becomes apparent. This issue will be discussed further in this chapter.

Rapid development of computer technologies and their software capabilities to support virtual co-location CE provide a viable alternative for a successful product design innovation which is well beyond what is possible with current practice.
2.2.2 Factors Affecting the Changes

(a) Competitive Environment

The emergence of Japan as a superpower in manufacturing industry had opened the eyes of the industrialised countries. Japan had emerged as a major new competitor in almost every aspect of manufacturing industry. The unique organisation culture that the Japanese possess seems to contribute and help them master current technologies ahead of their competitors and their ability to structure these technologies to fit their unique environments have benefited them. As reported by Hitomi (1993), from 1950 to 1990 Japan’s real gross domestic product increased almost nine times, the output of manufacturing in monetary value increased seventeen times, and the value of this industrial sector increased twenty-one times.

Prasad (1996) reported that when research on CE was carried out, the Japanese had already implemented the CE concept. He supported the statement by a study done in Europe that compared the time to market for automotive products between Japanese and European manufacturers. Based on eleven projects that the studies had carried out, Japanese companies could develop and introduce a new car to the market twenty months faster than the Europeans. The ability of Japanese companies to bring a product earlier to the market has made their products very competitive and able to sustain a longer time in the market. At least the product can be exported and be the first to be introduced to the market. By the time imitators arrive in the market, the producer can maintain the technology gap by continuous innovation.

Thus, the above two examples illustrated the impact of product innovation on competitive market. In order to achieve this aspiration, enterprises should be able to sense and interpret intelligently their market and environmental factors [Bennett et al, 1993]. Its organisational structures should be flexible to adapt both the customer requirements and competitive opportunities and able to transform those process innovations into technical success. Hence, the author believed that the introduction of
CE concepts in product design activities will help to improve product quality, reduce overall cost and time to market of a product.

(b) Tools Supporting CE

The effective implementation of CE was well supported by manufacturing management tools which were reported to advance in phase with the CE concept. Among key areas that contribute are Total Quality Control [Feigenbaum, 1991], Computer Integrated Manufacturing [Rembold et al, 1985] and Just In-Time Productivity Improvement [O'Neal et al, 1991; Hutchins, 1988]. Some key tools in each of these areas provide an enterprise with a competitive advantage that helps to reduce engineering and manufacturing costs, shorten the design and development time, reduce rework and thus significantly reduce the number of engineering changes.

- An important philosophy learnt from Total Quality Control tools is about how customer satisfaction was focused upon as their main objective. For instance in the Taguchi Method, Taguchi derived quality of a product as the losses imparted to the society from the time the product is shipped [Wu, 1991]. The Quality Function Deployment (QFD) [Overby, 1991] technique provides a framework for an integrated approach to design so that significantly more of the life cycle of a product gets appropriately focused upon at the beginning of the design process.

- In computer Integrated Manufacturing (CIM) where activities involved include Electronic data Interchange (EDI) [Bracket, 1994; Harris et al, 1996], Simulation and Analysis [Wetherbe, 1979; Senn, 1989], Networking and data communications [Goldman, 1995], group technology [Gallagher et al, 1986; Askin et al, 1993], Value Engineering [Lyman, 1992] and solid modelling to ensure the integration of manufacturing system as the current state of art.
Just-In-Time productivity improvement which include among other activities such as Design for Assembly (DFA) [Boothroyd, 1996], Design for Manufacture (DFM) [Corbett et al, 1992], Synchronous Manufacturing [Umble et al, 1990] and Continuous Improvement program [Ezop et al, 1989] is a business philosophy that focuses on removing waste from all the organisation’s internal activities and from external exchange activities [O’Neal et al, 1991].

(c) Advancement of Computer Technologies

The computer industries are changing dynamically where almost every three to five years a new family of computers is introduced to the market. As computers’ capabilities continue to increase their task-performance capabilities also increase. The potential applications of computers also increase and change in form. From an application approach where computers were used to solve specific problems and supplied specific information, they have moved to data integration applications. At this stage entry integration applications enable the transfer of data from one application to another. From here it has advanced to another stage where they are capable of performing distributed data processing [Pressman, 1994].

Today, computers are used in many industrial activities where they are capable of handling system integration for manufacturing processes. This integrated manufacturing system helps to manage information systematically where the same information is used and available when needed [Ulrich et al, 1985]. Synchronised information management eliminates duplication of work, makes data available upon request and maintains accuracy of information. This has made it possible for the data to be utilised in performing machine controls and logistic function where computer capabilities have made the feasibility of automating communication between machines and points the way towards an automatic factory.
The need to support the advancement in computer hardware has encouraged the development of peripheral devices. Networking technology is not an exception. Today it is common to find computers isolated physically but yet able to contribute effectively through efficient communication and information transfer. This development has benefited CE implementation, as it accommodates the CE concept of multi-discipline teams. A multi-site CE team can easily access design information from their offices. However, in manufacturing industries it is unusual to find a single supplier providing all the computing needs [Bowman, 1991]. In most cases, there will be a mix of mainframe, minicomputers, workstations and personal computers from a range of suppliers. Networking thus provides the integration in this multi-platform environment.

Another important challenge faced by computer developers is how to get these platforms to communicate and ensure the integrity of information transfer and manipulation. Open system architecture [Bowman, 1991] which was introduced in the late 1980s offered the solution to the above problem. It provides an operating system environment where computer systems and software from different vendors can be interchangeable and can be combined in an integrated environment. Standard operating systems like UNIX are now moving towards implementing this concept fully, and once successful should provide a conducive environment for interchangeability of different computer systems thus improving data accuracy and reducing throughput times.

The introduction of database management system (DBMS) [Widman et al, 1989] also helps to stimulate the use of CE. Its primary goal is to provide an environment both convenient and efficient in storing and retrieving information and allows large bodies of information to be managed efficiently. The newest direction in computer information systems research [Pfaffenberger, 1990] concerns the use of Artificial Intelligence integrated with databases.
2.3 CURRENT APPROACHES OF SOFTWARE DESIGN ON CE SYSTEMS

2.3.1 Current Approaches in Software Design

Understanding technology transition is necessary to understand and deal with software engineering [Walter, 1992], including programming languages, computer systems, data storage, life cycle and applications. Successful enterprises must plan and execute software technology transitions in such a way as to maximise customer satisfaction with the highest possible quality software products at the lowest possible cost of ownership. The rapid pace of technology advancement is pulling the computer industry into technology transition, observed in changes of programming languages moving from machine and assembly languages to fourth generation and to OO languages as illustrated by figure 2-3.

![Figure 2-3 Evolution of Software](Pressman, 1994)

As technology advances, computer scientist were looking for more efficient languages to support large scale programming, to reduce programming development
time. These programming languages featured top-down, more structured programming and encouraged modular design. The advancement in hardware technology like multi-processors that was introduced to the market, together with multi-platform concepts like open system architecture and networking have make it possible to support more powerful and efficient programming languages.

Today, programming languages like C and C++ [Waite, et al, 1993] are among the popular OO programming languages that are portable and can run on different platforms with little or no modification. These languages are flexible and focused towards the needs of the users. This latest discovery has contributed to research work on the OO paradigm that supports software developments which are easily maintained and where software components are reusable [Kaihara, et al, 1993]. OO design methods have evolved to help developers exploit the expressive power of object based and OO languages, using the class and object as basic building blocks. It is believed that this OO paradigm will contribute to the development of a Virtual Manufacturing Model in the future [Pressman, 1994; Booch, 1994] and will play an important role in creating virtual models and make the modelling operation much simpler and easier.

2.3.2 Present Software Design of Applications for CE System

A broad range of technology based research related to CE has been undertaken by academic institutions. An extensive literature review on computer aided CE (CACE) systems was reported by Molina et al (1994b). These activities have focused largely on developing software applications to support the implementation of specific process improvement techniques and integration frameworks that allow capture and sharing of cross-functional information. The integrated environment for this CE is achieved through the advancement of Information Modelling and amalgamation with decision support systems in an Information System Architecture.
The key implementation of CACE is modelling Engineering Information and most of the research work done so far is focused on Product and Manufacturing Modelling. Product Modelling [Gu et al, 1995; McKay et al, 1996; Krause et al, 1993] involved the modelling of information associated with a product and its components throughout its life cycle whereas Manufacturing Modelling [Young, 1994; Molina et al, 1995; Al-Ashaab et al, 1994] described the information modelling of manufacturing processes and resources in an enterprise.

The trends of this information modelling seem to move towards merging both models in an information system architecture. Applications are integrated within a framework and make use of the integration services of an information system architecture to access the information model. The development of the latter is further enhanced by the research work done in the area of computer-aided design support [Blount et al, 1995, Popplewell et al, 1995]. The architecture allows the integration of information and design support to be structured and transparent to the users. Among popular design support systems is the use of Artificial Intelligence [Blount, 1994; Faught, 1986; Grant, 1986; O'Keefe et al, 1987] to capture expert knowledge about the life-cycle of a product and use it to assist design activities.

While research work done on Product Modelling is moving towards a maturity, much effort is still needed in Manufacturing Modelling. Nevertheless it is just a matter of time before latter catches-up with the former. At present research has been successful in integrating the Product Model and Manufacturing Model but it seems to be lacking in offering feedback information about the response of the engineering system. Modelling tends to be rigid and focused on only the designer's opinion. The author sees this as an opportunity for research to be expanded beyond these capabilities and to include other elements in the enterprise.
2.4 POTENTIAL AREAS FOR FUTURE ENHANCEMENT TO CURRENT SOFTWARE DESIGN IN CE

The first phase of CE development has been successfully carried out and is mostly concentrated on designers' opinion or requirements. Even though there is still room for further improvement, the time will come for this to emerge as a mature piece of research. As more and more research is able to support the CE concept the time has come for researchers to explore the front-end activities and investigate the possibility of integrating marketing and finance modelling into present CE models. This will provide the manufacturing enterprise with a competitive advantage that will reduce engineering and manufacturing costs, shorten design and development, reduce rework and significantly reduce the number of engineering changes.

One of the most important up-stream activities, that has been recognised as an important element for any enterprise to sustain its business, is the ability to meet its customer requirements. The success or failure of the manufacturing enterprise depends on the need to have a clear mission and vision focused on the customer needs. The customer has a variety of sources to select from and the supplier who best satisfies the customer's exciting and expected requirements will be the choice of the customer [Clausing, 1994]. Perhaps it is not oversimplified to say that the best designed product is worthless unless there is a customer who wants its function and features and it can be manufactured and delivered within an acceptable time frame and price schedule.

Thus incorporating customer feedback into manufacturing generates dynamic performance benchmarks that further enhance the gathering of competitive intelligence and improve performance [Adiano et al, 1994]. A concept called Quality Function Deployment (QFD) [Akao, 1990] has been proposed to be included into the existing CE model.

The current CE Product Models capture information about a product based on perception from designers or engineers point of view. Anything perceived by a designer...
might not necessarily be accepted by the end users. Hence, input from end user is very important so that the product to be produced has a value added to it and is saleable. The QFD technique focused on this customer requirement and on gathering information necessary for determining what the customer truly wants. What the customer wants will determine whether new technologies are needed, whether simple improvements are possible or whether a revolutionary concept is required [Bossert, 1991]. The following section will describe the potential contribution of QFD if it can be integrated into virtual co-location CE.

2.5 OPPORTUNITY FOR INCLUDING QUALITY FUNCTION DEPLOYMENT (QFD) IN CURRENT SOFTWARE DESIGN FOR CE SYSTEMS

QFD is a comprehensive quality system aimed at satisfying the customer [Mazur, 1994]. The structured approach, focusing on customer priority by seeking out spoken and unspoken needs, has made this technique an excellent tool to deliver design values and translate these into actions and designs. QFD introduction into current management has benefited those implementing them [Zairi, 1993; Sullivan, 1986a].

2.5.1 QFD Theoretical Concept

(a) Definition

It is reported [Lo et al, 1994; Zairi, 1993], the term Quality Function Deployment is derived from six Chinese characters with Japanese pronunciation: hin shitsu that means qualities, features or attributes, ki no which means function and ten kai which means deployment, development or diffusion.
Chapter 2

There are several definitions of QFD reported in literature, and the significant keyword highlighted in most of the definitions is customers' needs that must be satisfied and translated throughout the product development process.

Among the earliest work reported about QFD is the work done by Kogure et al [1983]. They referred to QFD as activities needed to ensure that customers' required qualities are achieved by deploying quality related job functions step by step with a series of both objectives and means, down to the finest detail. Akao [1990] also defined QFD as a system of converting the customers' demands into design quality characteristics and developing a design quality for the finished product by systematically deploying the relationships between the demands and characteristics, starting with the quality of each functional component and extending the deployment to the quality of each part and process.

There are three important components of QFD methodology that were stressed by Akao: converting customers' demands into quality characteristics, developing a design quality for the finished product and systematically deploying the relationship between the demands and quality characteristics. These three components must be satisfied during the QFD process. The system approach concept as popularised by Akao was well supported by other authors [Sullivan, (1986b); Parrot, (1995); Daetz, (1989); Overby, (1991); Bossert, (1991)]. This approach described the integration of various quality tools used to generate the QFD matrix throughout the product development process.

There are also authors who described QFD as a structural planning tool that is used to ensure the voice of a customer is deployed throughout the product planning and design stage. Among them are Wasserman (1993), Halbleib et al (1993) and Hauser et al (1988). They argued that QFD provides the means for interfunctional planning and communications that focuses and co-ordinates skills within an organisation first to design, then to manufacture and market goods that customers want to purchase and will continue to purchase. Brown (1991) refers to QFD as a quality assurance system that
helps to ensure that the voice of the customer is clearly heard and followed in the
development of a product, through its structured application of four strategic concepts:
preservation of the voice of customer, a cross functional team, a CE approach and a
concise graphical display.

QFD, as defined by some authors [Evbuomwan et al, 1994; Brown, 1991;
Zairi, 1993], established a great opportunity to make CE work. The overall concept that
provides a means of translating customers' requirements into appropriate technical
requirements for each stage of product development and production create a customer
supplier chain. A system of converting customers' demands or means of translating
customers' requirements is actually an integrated tool that records customers' requirements and technical specifications in the form of matrices. This recorded information throughout the product development and design process is transparent to all members of the cross-functional team involved in the process.

The following section will described the implementation QFD methodology
that has proved very successful and benefited the manufacturing enterprise.

(b) QFD Approaches

Several different approaches to QFD have been developed over the years. However, it has been recognised that none of the approaches is perfect and indeed should be tailored to the situation which multi-disciplinary design team members find more comfortable to use. Hales (1993) reported two QFD approaches that are currently used namely the Matrix of Matrices that is promoted by GOAL/QPC and the Four Phase promoted by American Supplier Institute (ASI). The Matrix of Matrices approach consists of 30 matrices which describe everything from how to convert customer requirements into product characteristics to how to prioritise Failure Mode Effects Analysis (FMEA) studies. However, Hales concluded that this approach often scares

---

GOAL/QPC is a non-profit organisation based in Methuen, Massachusetts, USA
new users because of the transaction of the 30 matrices and it does not lay out a process which is conceptually easy to follow. Brown (1991) reported that a more popular approach to QFD, thus promoted by ASI which consists of four QFD matrices. The structured approach of the QFD matrices, transforms the voice of the customer at every phase of the product development cycle and provides the understanding and degree of involvement that the design team could contribute. This simple structured approach, explained in the next section, is adopted in this research.

(c) The Four Phase Approach

The Four Phase approach is satisfied through a series of charts and inter related matrices that deploy customers' requirements from product planning and design planning to process planning and production planning. The four phases are: planning, design, process and production. The QFD methodology is shown in figure 2-4.

Figure 2-4 The Four Phases of QFD Process [Lo et al, 1991]

As illustrated in figure 2-4, the four phase matrices implement a strategy that uses selected design requirements from product planning as the input to the left column of
the action requirements named as second phase matrix. In third phase, selected action
descriptions from action deployment is used as inputs to the left column of the process
planning and in the final phase, manufacturing operations from the third phase is used as
inputs to the left column of this final phase [Lo et al, 1994; Brown 1991; Hauser et al,
1988].

(d) THE HOUSE OF QUALITY (HoQ)

The first phase of the QFD methodology is the product planning phase. This
phase is very important as it provides the foundation of the remaining phases of QFD
implementation. It starts with listing the customers' requirements and transforms these
requirements into product features and functions or design requirements. The tool
associated with this first phase is called the House of Quality (HoQ) [Hauser et al, 1988]
or the quality table. The term HoQ is commonly used since the shape of the graphics
resembles a shape of a house segmented into eight different compartments as depicted
in figure 2-5, and explained below.

![Figure 2-5 The House of Quality (HoQ) (Hauser et al, 1988)]
Voice of the customer (The Whats) and Customer Importance

The construction of the HoQ begins with the customers' needs obtained from marketing and research activities. These requirements are recorded in this compartment, often called the 'whats items'. For easy understanding and ease of operation this compartment is sometimes divided into two or three sub-compartments, the primary, secondary and tertiary needs. The most important step in implementing HoQ is to define customers' needs. If these are not defined correctly analysis of the subsequent information will not be accurate, since the whole analysis is based on the discovered needs. The process of discovering customers' needs shall include [Mazur, 1994] being a customer, studying customer behaviour and communicating with customers or simulating their behavioural patterns. Information regarding product can also be derived from warranty data or complaints, customer reports and company personnel especially from marketing and sales departments. During information gathering customers' importance or priority for each need is defined. Customers' needs are ranked and rated in this compartment.

Engineering Characteristics (The Hows)

Customers' needs are typically subjective expressions helpful in developing the understanding of what the customers want but offer little guidance about how to design and engineer the product [Ulrich, 1995]. In this compartment, the development team establishes a set of specifications that signify precise measurable details of what the product has to do. Ulrich emphasised that product specifications do not tell the development teams how to address the customers' needs but they do represent an unambiguous agreement on what the teams will attempt to achieve in order to satisfy the customers' needs. Information about technical design requirements represented in this compartment consist of a precise measurable product description together with their measurement units. Similarly, to ease the process of defining product specification this compartment is also divided into two or three sub-
compartments, the primary, secondary and tertiary depending on the clarity of the process.

☐ **Relationship Matrix**
The relationship matrix is the core of the HoQ. It relates the enterprise to the customers by systematically matching the customers’ needs with the design specifications. A relationship is established to check whether there is a very strong, medium or weak relationship, indicating how much each product specification affects each customers’ need. It is usually denoted by symbols.

☐ **Correlation Matrix**
Evaluations of design specifications are done using the correlation matrix. The specifications are matched against each other and correlation are established to specify the effect on each specification of changing any of the other specifications. Correlations are ranked as strongly positive, positive, none, negative and strongly negative denoted by symbols.

☐ **Competitive Benchmark**
The satisfaction of customers’ needs from competitors’ products as perceived by the customers are recorded in this compartment. This will complete the horizontal relationship of the HoQ. Strong points about the products obtained from this analysis are an asset to marketing teams that can be used as advertising features. Improvement should be made if there are any weaknesses identified from the comparison. Further enhancement to this competitive benchmark includes sale points that allow the marketing team to rate whether or not they would get leverage out of any improvements [Shillito, 1994].

☐ **Technical Importance**
The compartment immediately below the relationship matrix represents the importance ranking of the product specifications. The assessments of the
product specifications that include technical difficulties, development time and cost are done in this compartment. Teams members use this data as a reference point to decide the efficiency of various technical solutions.

- **Technical Competitive Benchmark**
  Information about competitive products is important to gauge the competitiveness of the product specifications. Gathering information about competitors products is usually time consuming as it involves purchasing, testing, disassembling and estimating the production costs of the most important competitive product [Ulrich, 1995]. However, this investment of time is essential as the success of the product specifications’ evaluations will be dependent on these information.

- **Target Values**
  This compartment highlights the target values of the product’s specifications established in the Engineering Characteristic compartment. Competitive target values are obtained by comparing the present values with other competitors available in the technical competitive benchmark compartment. The target products’ specifications may be used to generate the next phase of the HoQ.

### 2.5.2 Benefits of QFD

The concept of QFD is two fold. QFD targets the customers' needs as its priority. While assuring the needs are translated throughout the product development process, QFD enriches intercommunication between team members of different departments. The proactive preventive’s approach [Lo et al, 1994] introduced by QFD methodology identifies customers’ needs and translates those needs into technical requirements. Problems and discrepancies in the product development and design process are brought to surface earlier before the drawings and specifications of the products are produced.
Chapter 2

The process produces competitive and reliable products and reduces product development cycle and start-up cost.

Sullivan [1986a] reported that QFD has been used by the Japanese automobile industry giant, Toyota. In 1979 two years after launching their new van, Toyota obtained a reduction of 20% in their start-up cost. They obtained a further reduction of 38% in 1982 and a cumulative 61% reduction in 1984. During this period the product development cycle was reduced by one third with a corresponding improvement in quality because of a reduction in the number of engineering changes.

Other authors that observed encouraging results obtained by companies who used QFD technique include King (1989) and Zairi (1993). King reported from a study conducted in Japan, design time was reduced by one third to half. He found out that companies using QFD have been able to set and plan design quality more easily, information about competitors’ products were made available and communication between multi-disciplinary team members is achieved. Zairi (1993) also attributed similar success to companies adopting QFD. They acknowledged that they needed few design changes, acquired shorter product development time and fewer start-up problems, the team members received transfer of knowledge and their customers were satisfied.

In short, the bottom line of QFD implementation is higher quality, lower cost, shorter time to market and a substantial marketing advantages.

2.5.3 Potential Reinforcement of Current QFD Implementation

QFD is a complex system that requires companies to adopt new and different methods of doing things, and new and unfamiliar ways of working together, hence making the successful application of QFD a challenge to even the most progressive
organisations. Zairi [1993] reported some of the problems faced by companies who had experienced using QFD.

Another main problem is the implementation of QFD matrices, the HoQ. Daetz [1989] reported that because of the large size of the QFD matrix, insufficient clerical help, and no suitable matrix display software. The team never saw their HoQ matrix in one piece. Hanson [1993] also reported a similar problem where he experienced an uphill task to start the QFD process due to the lack of software packages. Charts and calculations were generated by hand that were both distracting and cumbersome. Supporting evidence hinted that as QFD was originally implemented as a set of charts and procedures, these manually recorded paper forms have certain serious limitations [Wolfe, 1994]. Syan et al (1994) concluded:

"Overall, QFD is currently completely manual or is aided, via a static documentation process by computer system. This greatly restricts the potential of QFD as defects in design parameters from Computer Aided Design systems or databases cannot be automatically detected nor their consistency automatically checked".

The introduction of commercially computer supported packages as reported by Vora et al, [1989]; Fowler [1991]; O’Conner et al, [1992] and Foutz [1993] to replace manually recorded QFD charts perhaps has partially eased the existing problem and has proved to be successful for small systems. It is believed that the introduction of more sophisticated commercially supported computer packages in the future will ease the implementation of QFD technique. This may be an interesting aspect of implementing QFD technique with regard to larger and more complex system. There is work done by Wolfe [1994] which uses Decision Support System with hypertext concept to mitigate the above problem. Oliver [1990] has been reported to use knowledge based system, Sriraman et al, [1990] uses OO databases while Bird [1992] adopted an Expert System to enhance QFD. Another interesting piece of work is the association of CE with QFD as reported by Krishnawamy et al, [1992].
It is believed that the use of QFD will continue to be developed as it matures as a tool for product development and design and as investigations wish to examine more complex systems.

2.6 SUMMARY

The dynamic and rapidly changing behaviour of the business environment today has forced manufacturing enterprises to realign their business strategies to be in phase with their competitors. They are now searching for alternative means in order to be able to survive in this competitive world. CE implementation as described has been seen as an effective supporting tool towards achieving this goal. The enhancement of CE implementation by integrating computer technology is a timely effort to take advantage of its enormous capabilities. Potentially this should bring benefit to product innovation and at the same time reduce product life-cycles. The foregoing literature review has considered this contemporary CE implementation and the model of this manufacturing information in the form that could be processed by computers.

The modelling of these manufacturing information systems in structured Product and Manufacturing Models has laid the foundation for further research. Choice for further investigation should include upstream activities as these have proved to be vital for the survival in the new competitive business environment. The author believes that the investigation will help an enterprise to get an initial product early to the market. Thus integrating QFD is precisely the correct ingredient to enhance the current CE system. This integration is expected to stimulate important developments that will contribute competitiveness in product design activities.

The next chapter will described the methodology conceived by the author and how the research was carried out.
3.1 INTRODUCTION

Chapter 2 has considered major problems which constrain current CE practice in product design and the experience of modelling the engineering information as a practical solution. It has identified QFD as a tool that will enhance the engineering information model.

This chapter will establish a methodology that describes the QFD information modelling process. This includes identifying a suitable modelling structure and database that can model and store the QFD information. The solution will also allow the implementation of knowledge based systems as decision support tools that enable QFD information to be analysed and managed.

3.2 ISSUES INVOLVED IN DESIGNING A QFD INFORMATION MODEL

The literature survey illustrated the need to find new approaches that will enhance CE systems so that products are produced that suit the customers' needs with less
development effort over shorter lead-times. In seeking to achieve this, the literature also highlights the need to integrate upstream activities that will stimulate a competitive atmosphere and improve the product design process. QFD has been established as providing a mechanism that could achieve the above objective. In addition, the literature emphasises the need to represent the QFD information in a computer environment to explore the potential computer technology advancements that will provide an added value to the product being designed.

Based on the established facts described in chapter 2, and drawing on the lessons learnt, it has become evident that the following key issues in modelling QFD information need to be addressed.

3.2.1 Modelling QFD System

QFD methodology as described in chapter 2, is indeed a complex process [Frew et al, 1993] and if applied to a large system in its original form, its potential ability to guide the design process is expected to be less efficient. Furthermore, in order to accommodate the increase in demands of product introduction and to offer a wider range of variants, effort must be focused on shortening product design lead-times and using flexible design tools [Bennett et al, 1993]. This includes modelling QFD information that represents an abstract of the QFD process. This information model that represents the inherent complexity of a large system should be able to facilitate the following characteristics.

(a) Modularity

In the context of this thesis the term modularity will be used to describe the decomposition of a system into a set of cohesive and loosely coupled modules. Modularity allows a high degree of independence among modules, and communication
between modules is performed using procedural interfaces. This act of partitioning a program into individual components reduces complexity. The decomposition of the program into smaller modules enables software to be designed and revised independently thus permitting a simple structure that is easy to comprehend. Any changes needed in a module can be carried out without affecting the behaviour of other modules, hence reducing recompilation cost of the affected module.

Modularity also helps software components to be managed and maintained systematically. This capability distinguishes the concept from traditional software programming where information system applications have undergone many generations of changes and are now virtually unmaintainable [Pressman, 1994]. Even the smallest modification can cause the entire system to fail. Modularity permits manageable and maintainable documentation of software that enables efficient use of memory and optimises program execution speed.

(b) Reusability

Reusability is an important characteristic of a high-quality software components [Pressman, 1994]. Software components represented in the QFD information system are designed and implemented in such a way that they can be reused in many different programs. These reusable software components encapsulate both data and processing in a single package enabling the software to support the creation of new applications from reusable parts. For example, in today's computer aided design packages most of the standard components such as dimension symbols are contained within a library and are available when required during design.

Application of this reusability concept is a powerful feature found in the OO programming paradigm that is becoming popular in today's state of art programming tools. The OO technology concept provides the capability of improving software reuse and enhances its capability by adopting sound principle and experience found in older
programming methods [Booch, 1994]. It encourages the reuse of not only the software but entire designs, leading to the creation of reusable application frameworks.

3.2.2 Database Management System (DBMS)

In the previous section, the author has established some characteristics of programming that will be adopted to develop the QFD Information model. OO methodology has been identified as a tool that facilitates these characteristics. In OO methodology, the universe is viewed as a collection of independent objects that communicate with each other by passing messages. Each object contains its own data and methods, and appears to have a life of its own, interacting with other objects to create a system.

Data embodied in the object is likely to be managed most efficiently when stored in a database as persistent objects. These persistent objects continue to exist outside the scope of program execution and retain all their object properties, providing in this context the capability of retrieving QFD information as required by application software. This has led the research to require a database system having the following basic characteristics:

- Should be able to facilitate instantaneous data access to required QFD information.

- Should be able to facilitate the safety of information despite system crashes or attempts at an unauthorised access. Database should also facilitate smooth recovery ability that will allow database to survive hardware and software failures in a coherent and consistent state.
Should be able to avoid anomalous and redundant data as the data is shared by multi-discipline team. It should also support complex operation of the multi-discipline team in multi-platform environment.

- Should be persistent to allow the database to survive the termination of a process so that it can be used by another process.

- Should support concurrency that will allow multiple users to share data simultaneously. Hence, this will enable a highly parallel, co-operative application development process.

- Should have an efficient and independent query capability that allows an application programmer to access specific information in the database.

- Should be able to support complex data structures and relationships that enable development of QFD information to have complete freedom to define, create, delete and modify data structure.

The characteristics identified led the author to adopt database system to store captured QFD information in a single data repository. The DBMS regulates data access in the shared database and provides the functions discussed so far. In addition, the DBMS manages and maintains the stored data such that it removes structural and data dependency of the system. Availability of the DBMS has encouraged the development of modelling the database known as data model in order to enhance its potential capability. In order to support programming application described in the previous section, OO data model has been adopted in this research. The discussion about the OO data model and how it is different from traditional DBMS can be found in Ozkarahan (1990), Vista (1989) and Rob et. al (1993).
3.2.3 Integratability

The QFD information system consists of many elements, including models and applications. These elements and applications may be distributed over many computing platforms and probably located at several sites. Thus the selection and use of a suitable architecture and a modelling method is necessary to support these integration requirements. The architectural framework should enable all elements and applications to work together in an attempt to provide an efficient communication.

There is a need to study the way in which QFD information flow in multidiscipline team is integrated and to map its environment within a virtual simulated environment. This is considered important because it can be very difficult and expensive to physically achieve such an integration. This means the actual stages of information capture should be precisely modelled.

3.2.4 Graphical User Interface (GUI)

A graphical user interface (GUI) will be proposed as an interfacing application package to the above information modelling system. The primary goal of a user interface is to help designers easily create applications that increase user effectiveness and satisfaction and thus avoid laborious tasks in getting the application running. However, since the development of the GUI is beyond the scope of this research, the author has attempted to modify and utilise an existing GUI to meet the requirement of this research.

3.3 STRUCTURING RESEARCH REQUIREMENT

Figure 3-1 illustrates the methodology adopted by the author in addressing the issues raised in the previous section. The following section will discuss the steps
incorporated in the methodology that formed the main activity of the research and which reflect in the structure of the thesis.

3.3.1 Analysis and Selection of Architecture

An architectural model of the QFD information will typically describe interrelations among the essential elements of the QFD system. In choosing a suitable architectural model for the study, the author analysed the capability of existing information system architecture that can facilitate QFD information architecture under development. Research found related to information modelling [Bugtai et al, 1997; Harding et al, 1996a; Molina et al, 1994a] for the support of CE has been primarily concerned with Product and Manufacturing Modelling.
It is generally accepted that Product Modelling involved modelling of information associated with a product and its components throughout its life cycle whereas Manufacturing Modelling described the information modelling of manufacturing processes and resources in an enterprise. Since the first phase of the QFD process involved capturing information about design features and related product design activities, the QFD information model will reside as part of the Product Model. The architecture of the QFD information model will be based on the following essential characteristics:

1. Well structured and documented information architecture that is capable of modelling large system with inherent complexity.

2. Ability to support the re-use of software components.

3. Ability to facilitate modular task structure.

4. Flexibility of the solution generated that will not constrain system development.

5. Generic and not limited to specific product. It can be used to populate other systems.

6. Strong database system that provide information storage and facilitates integrity of the captured information.

The author's investigation of available QFD information architectures led to the conclusion that no single available architectural model provided the support of all the essential characteristics listed above. Hence the main objective of this research is to establish QFD information structures within a Product Model to support product development activities.
Increasingly, today’s product development environment needs to be responsive to change. The introduction of QFD information within a Product Model is expected to show clear design benefits, such as a more adaptable product design approach responsive to customers’ needs, more effective use of multi-discipline resources and well managed communications within the multi-discipline team.

3.3.2 Selection of Customers’ Needs and Design Features

The first most important step of the QFD technique is to define customers’ needs. If these are not defined correctly everything that follows will be wrong, since the whole analysis is based on the discovered needs. Techniques that described the compilation of customers’ needs and their relative importance can be found in much of the literature [Mazur, 1994; Griffin et al, 1992; Ulrich et al, 1995]. In this research, it is assumed that the customers’ needs are already documented. The research provides an information architecture that captures this information and later guides the multi-discipline team to define what remains subjective about specifications for a product.

The information architecture allows customers’ needs to be captured in a hierarchical structure of primary and secondary needs. This allows the design team to define strategic direction for the product as its primary needs and identifies more specifically what the design team should do in order to satisfy the corresponding primary needs. The implementation of the hierarchical information structure is based on experience of discussion with industrial users and strong recommendation from literature [Shillito, 1994; Griffin et al, 1992; Morell, 1987].

The same concept is applied to the design features where a hierarchical structure is also used to capture design features information. The captured design features information is categorised into aspect and specification. Hierarchical structure provided by the information model is expected to ease the implementation of capturing QFD information.
3.4 CONSTRUCTION OF THE DECISION SUPPORT SYSTEM FOR THE QFD INFORMATION MODEL

As established in Chapter 2, QFD is a complex system and requires an enterprise to adopt new and different methods of doing things, as well as new and unfamiliar ways of working together. This has made a successful application of QFD a challenge to even the most progressive organisation. The development of the QFD information model as described in previous sections is envisaged to ease the implementation of QFD process.

In a conventional QFD system, the HoQ helps the team to set target specifications entered on the bottom line of the HoQ. The HoQ encourages multi-discipline design teams to work together to understand each priority and goal but the HoQ does not relieve any team members from the responsibility of making decisions. It does facilitate the design team in debating priorities. In a small system, it is simple and easy to complete the HoQ information and define the target specification for each design feature. However, as the magnitude of the QFD matrix increases, to capture and analyse the system efficiently becomes tedious and time consuming. The information structure proposed earlier addresses the problems raised but the analysis of the QFD system remains problematic if the software only supports the storage of the captured information.

This understanding has led the author to explore the possibility of enhancing the information architecture to include Artificial Intelligence (AI) capabilities. Hence, the sub-goal of this research is to demonstrate intelligent analysis of QFD information contained in a Product Model. AI techniques will be exploited and structures that accommodate the ability to manage the QFD information will be developed. The implementation of this concept is discussed in chapter 5 of this thesis.
3.5 EXPERIMENTAL ENVIRONMENT

3.5.1 The Model Oriented Simultaneous Engineering System (MOSES) Architecture

The software environment adopted in this work is based on an architecture developed by a previous research project (MOSES\textsuperscript{1} [Harding et al, 1996a]). The MOSES concept is based on the use of two information models, a Product Model and a Manufacturing Model that can be accessed by an open set of application programs via an integration environment [Harding et al, 1996a] as depicted by figure 3-2. Both the models together with the human designer are considered as design agents that contribute their knowledge and expertise in developing a product design.

![MOSES Computational Viewpoint](image)

The proposed QFD information model, to be included as part of the Product Model, fits well within the MOSES architecture. The MOSES architecture facilitates the introduction of multiple loosely coupled applications to support a variety of design

---

\textsuperscript{1} The MOSES project was carried out by Universities of Loughborough and Leeds in collaboration with several industrial and commercial organisations. It was funded by EPSRC under grant reference GR/H 24273.
disciplines as depicted by figure 3-3. Thus a QFD system can easily be added to the computer aided engineering environment.

Figure 3-3 The MOSES Architecture and Integrated QFD

The MOSES architecture also ensures that all the design agents are able to communicate and negotiate at what point they should be contributing. Figure 3-4 illustrates the interaction activities of a typical application using the QFD information model.

Figure 3-4 Using the QFD Information Model
An Engineering Moderator (EM) [Harding et al, 1996a] acts as an intelligent monitor of the design activity, and is an active driver of necessary collaboration between design agents. EM is a specialist manager or co-ordinating program whose role within MOSES architecture is to drive concurrency [Mc Kay et al, 1995]. The structure of the EM consisted of three expert modules that are implemented using an OO design approach. These experts modules are instances of a Knowledge Representation Model (KRM) [Harding, 1996b], enabling the EM to achieve the following objectives:

(a) Detect changes or conflicts occurring in the product design.

(b) Identify and inform interested design agents about the detected change.

(c) Monitor interested agents to ensure resolution of possible conflicts of interest.

3.5.2 Programming Language

The QFD information model was developed using OO methodology. In OO methodology the essential characteristic of an object is abstracted to distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer. An object has attribute, state and behaviour that represents the real-world entity. Figure 3.5 illustrated an example of abstraction of a customer needs written in C++.

```cpp
class NEEDS_aro
{
  private:
    char the_need[200];
    int NEEDS_importance;
    int NEEDS_relativeImportance;

  public:
    NEEDS_aro(char* needname);
    char getthe_need(void);
    void put_the_need(char* need);
    int getNEEDS_importance(void);
    void putNEEDS_importance(int important_rating);
};
```

*Figure 3-5 The Class Structure of NEEDS_aro*
Objects communicate via messages sent to each other, with the assistance of the underlying language and operating run time support systems [Lamia, 1995]. Each message received by an object should have a procedural code that interprets and carries out the function requested by the message. For instance, as shown by figure 3-5, an object is created as an instance of NEEDS_aro class via the constructor NEEDS_aro. Similarly, methods such as put_the_need and putNEEDS_importance will invoke messages that carry values to define the object attributes the_need and NEEDS_importance respectively.

Messages are the only interface that the object presents to the external world. Any information or internal state of the object cannot be directly manipulated. This is only possible by means of methods offered in the accessible interface of an object. Denial of access to the structure ensures the integrity of the object's state and hides the object's internal details. This ability to hide the object's internal details is known as encapsulation [Rob et al, 1993]. Thus, it is possible to design internal object representation of the information and procedure in any manner as long as the message interface remains consistent.

Another key element of the OO methodology is its ability, known as Inheritance, to facilitate the abstraction of common characteristics of objects into a tree like structure and avoid redundant descriptions of data and procedure attributes. Inheritance allows object classes to be assembled into a hierarchy of classes and subclasses. Each subclass inherits the characteristic of its superclass and will automatically inherit any changes that are made to its superclass.

OO methodology also promotes the use of Polymorphism, that is the term used to describe two different methods that have the same name, but perform different processes in different objects [Lamia, 1995]. For example, objects of classes Circle, Rectangle and Square all understand and respond to the method findArea but the algorithm for performing the calculation is different for each class.
3.5.3 Database System

The KRM (Chapter 5 of this thesis will discuss the KRM in greater detail. Further reference can be found in Harding [1996]) together with the Product and Manufacturing Models reside as elements of a single federated OO database to minimise the overhead of computing resources. The database used for this research is Objectivity/DB. The author believed that potentially such database systems should provide very powerful technology and facilities. Objectivity/DB is described to be a high-performance, OO DBMS for engineering and commercial applications, ideally suited for applications that require flexible data modelling. This data modelling involves complex relationships and demands high performance. Objectivity/DB provides full database functionality with a completely distributed architecture that supports multi-platforms and is designed to manage large amounts of data transparent to both the developer and the end user. In order to facilitate interactions with the database structure the application will be written in C++.

Objectivity/DB uses four logical storage entities known as Basic Objects, Container Objects, Database Objects and Federated Database objects. The fundamental unit of storage is the Basic object. Each Basic object belongs to a single Container object, each Container object belongs to a single Database object and each Database object belongs to a single Federated Database object as shown in figure 3-6.

![Objectivity/DB Logical Storage Model](image-url)

**Figure 3-6 Objectivity/DB Logical Storage Model**
3.5.4 Motif Environment

The GUI for the application software is achieved by utilising Motif package provided by Open System Foundation (OSF) available in the SUN workstation. In Motif, the toolkit of “pre-programming window” is the central feature that can be used to structure programming windows. It supplies the base elements from which user interfaces can be built. The toolkit can be separated into three different layers, each supported by a separate library [Berlage, 1991].

3.6 RESEARCH FOCUS

Consequent to the literature review about the CE activities in product design, the author decided to approach his research study as depicted in figure 3-7. The focus is on providing information architecture to support modelling of QFD information and integrating the QFD information into product development activities. This will help to enhance the product development process and hence provide a competitive edge for an enterprise to excel.

Hence the author seeks to explain how the methodology is implemented in the forthcoming chapters. This could impact practice in three ways,

- it provides a method of capturing QFD information within the Product Model, accessible for the design team to manipulate design features based on the customers perspective;
- it provides a structured and organised communication platform for multi-discipline design teams to make decisions;
- it provides a basis for computer assistance in analysis of QFD data;
Chapter 3

Figure 3-7 Research Focus and its Environment

Properties of Concurrent Engineering system and its compatibility with QFD system

Concurrent Engineering System

Artificial Intelligence Technology

Knowledge Base

Fuzzy Logic

Neural Network

QFD System

Architectural Framework & Modelling Technology

Graphical User Interface (GUI)

Database Management System

Object Oriented Programming

Operating System

Research Requirements

Faster time to market

Reduced development cost

Encourage communication

Well documented information

Fewer design changes

Re-usable Software Components

Development of QFD information Architecture and Software

Open system Architecture

Research Impact on Practise

Method of capturing QFD Information within Product Model

Basis for computer assistance in analysis of QFD data

structured communication platform for multi-discipline team

53... Research Methodology and Procedure
3.7 SUMMARY

The thesis will focus on two important aspects:

☑ QFD founded upon a Product Model that allows all design agents access to QFD information throughout the design process.

☑ Intelligent QFD experts to accomplish useful, consistent, reasoned analysis of QFD information.

The emphasis of work is on enabling the design of a reusable and modular information structure.
4.1 INTRODUCTION

This Chapter demonstrates a novel methodology which formally structures the QFD information within a Product Model. Once QFD information is established in the product model, it enables customer attributes to be stored and associated with a particular product. The approach described here uses Booch OO design methodology (Booch, 1994).

In this chapter an explanation on how to use essential features built as part of the software components in the QFD information model is provided. It includes the use of a graphical user interface and database browser. The aim is to highlight its ability to facilitate easy communication in a friendly environment between the application software and the end-user. Key parameters used as attributes of the components in the HoQ will also be described.
4.2 SELECTION OF QUALITY FUNCTION DEPLOYMENT (QFD) INFORMATION ARCHITECTURE

4.2.1 Background

As described in Chapter 2, QFD methodology is a disciplined approach that provides an efficient environment for design team members to interact and exchange information about the product design. Information created and required during the design process can be shared between design team members and the structures proposed here enhance this capability by adding QFD information to the Product Model. This gives CE team members the ability to handle customer requirements throughout the design process.

The information structure proposed in this QFD information system architecture has also been designed to enable the QFD information to be captured within the Product Model OO database. To achieve this, the Booch OO design method has been adopted (Booch concepts and definitions described in Chapter 3).

4.2.2 QFD Information Model Structure

QFD information used in product design and the suitability of the four phase approach adopted in this research has been described in Chapter 2. This chapter will described the structure implemented for the first phase, popularly known as the house of quality (HoQ). The research has been focused on implementing the structure for the HoQ, as the HoQ is the nerve centre and the engine that drives the entire QFD process [Shillito, 1994]. Matrices that form the HoQ as described in Chapter 2, are modelled. The 8 elements represented in the HoQ are listed below:

- The Voice of the Customer or the Customers’ Needs;
- Engineering Characteristics;
Chapter 4

- Relationship Matrix;
- Correlation Matrix;
- Competitive Benchmark;
- Technical Importance;
- Technical Competitive Benchmark;
- Target Value.

The structured segmentation of the HoQ, makes it possible to establish the class structures using Booch OO Design Methodology. This information structure that described the captured QFD information is depicted in figure 4-1.

A key class, PRODUCT, within the Product Model must be named and described. QFD analysis will show that there are multiple customer needs associated with the product, and which design features of the product might be identified as satisfying these needs. Hence the required information structures will demonstrate a one to many link between instances of the class PRODUCT and instances of the classes representing customers' needs and classes representing design features.

As described in Chapter 3, information about customers' needs are classified into sub-groups based on similarities between these needs. The information structure provides this mechanism by defining a PRIMARY NEEDS class as its first level and a NEEDS class as its sub-level. The PRIMARY NEEDS class shall have a many to one link with the PRODUCT class whilst NEEDS class shall have a many to one link with PRIMARY NEEDS class. Both classes have their own attributes that store information about the class object. The attributes include their description, their customer's importance and their customer's relative importance.

Similarly, the process of capturing design feature information is divided into two stages and captured using two related classes namely by ASPECT and SPECIFICATION classes. ASPECT describes the main characteristics of the design
feature whilst SPECIFICATION describes its detailed characteristics. An ASPECT class has a one to many association link with SPECIFICATION class. The process of capturing data about customer needs and design features is further described by Table 4-1, which shows a simple case study of a mug design undertaken using a colleague as a newly identified customer. As described by table 4-1, a description for ASPECT class CONTAINER has many SPECIFICATION attributes such as max. height, total mass, chemical reaction, capacity, material thermal conductivity, bottom concave angle and surface texture.

![Diagram of the QFD Information Structure](image)

*Figure 4-1 Class Diagram of the QFD Information Structure*
Table 4-1 An Extract of the House of Quality for a Mug

ASPECT and SPECIFICATION classes also have their attributes that store information about the class object. Table 4-2 illustrated elements of the class structure that contains class attributes, its associated class and the relationship between those with their associated class.

<table>
<thead>
<tr>
<th>CLASS</th>
<th>CLASS ATTRIBUTE</th>
<th>ASSOCIATED CLASS</th>
<th>ASSOCIATED LINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT</td>
<td>DESCRIPTION</td>
<td>PRIMARY_NEEDS</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASPECT</td>
<td>1:m</td>
</tr>
<tr>
<td>PRIMARY_NEEDS</td>
<td>DESCRIPTION</td>
<td>PRODUCT</td>
<td>m:1</td>
</tr>
<tr>
<td></td>
<td>PRIMARY_NEEDS_IMPORTANCE</td>
<td>NEEDS</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>PRIMARY_NEEDSPEC_relativeImportance</td>
<td>PRIMARY_NEEDS_SPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>PRIMARY_NEEDSASCPECT_relativeImportance</td>
<td>PRIMARY_NEEDS_ASPECT_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td>NEEDS</td>
<td>DESCRIPTION</td>
<td>PRIMARY_NEEDS</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>NEEDS_IMPORTANCE</td>
<td>SPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>NEEDSPEC_relativeImportance</td>
<td>COMPETITOR_NEEDSCORE</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>NEEDASPECT_relativeImportance</td>
<td>COMPETITOR_NEEDSCORE</td>
<td>1:m</td>
</tr>
<tr>
<td>ASPECT</td>
<td>DESCRIPTION</td>
<td>PRODUCT</td>
<td>m:1</td>
</tr>
<tr>
<td></td>
<td>ASPECT_IMPORTANCE</td>
<td>SPECIFICATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>ASPECT_TARGET</td>
<td>PRIMARY_NEEDS_ASPECT_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>ASPECT_DIFFICULTY</td>
<td>ASPECT_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td>SPECIFICATION</td>
<td>CONTEXT</td>
<td>ASPECT</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>QUANTITY</td>
<td>SPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>QUALITY</td>
<td>SPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>MEASUREMENT_UNIT</td>
<td>COMPETITOR_SPECVALUE</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>SPEC_IMPORTANCE</td>
<td>PRIMARY_NEEDSPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>SPEC_DIFFICULTY</td>
<td>PRIMARY_NEEDSPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
<tr>
<td></td>
<td>SPEC_TARGET</td>
<td>PRIMARY_NEEDSPEC_INTERRELATION</td>
<td>1:m</td>
</tr>
</tbody>
</table>

Table 4-2 Elements of QFD Information Model Class Structure for PRODUCT, PRIMARY_NEEDS, NEEDS, ASPECT and SPECIFICATION Class
Each design feature (ASPECT or SPECIFICATION) is associated at least one customers' need (PRIMARY_NEEDS or NEEDS) which describes the product in measurable terms and should directly affect customers' perception. If it does not affect any customer needs then it may be redundant or the designer may have missed the customer’s need. The designer may also indicate the degree of technical difficulty and show how easy or difficult to make changes.

The four defined classes, PRIMARY_NEEDS, NEEDS, ASPECT and SPECIFICATION, determined classes that interact between them. For instance, interaction between customers’ needs and design features leads us to define classes that will reflect this interaction. Hence, a class PRIMARY_NEEDS_ASPECT_INTERRELATION class is available to create interrelationship matrix instance between the interaction of PRIMARY_NEEDS and ASPECT. The Interrelationship Matrix between PRIMARY_NEEDS and SPECIFICATION is created by an instance of PRIMARY_NEED_SPEC_INTERRELATION class. Similarly, the Interrelationship Matrix between NEEDS class and ASPECT is created by an instance of ASPECT_INTERRELATION class and NEEDS and SPECIFICATION by instance of SPEC_INTERRELATION class respectively. These classes supply information about relationships between customer needs and design features. A value given for this interaction indicates the strength of the relationship.

Design features of a product should be improved coherently (ie. not improving one feature at the expense of others) and this is achieved through the correlation matrix as described in Chapter 2. Thus, creating correlation matrix as a class will provide information to facilitate necessary engineering trade-off. Two important design features classes, ASPECT and SPECIFICATION defined another two correlation classes the ASPECT_CORR class derived from ASPECT class and SPEC_CORRELATION derived from SPECIFICATION class. The link between these classes are described in table 4-3.
Another important class in the proposed information structure is the COMPETITOR class that captures customer input about competitors' products and provides benchmark about the product in the market. From this COMPETITOR class, COMPETITOR_NEED_SCORE, COMPETITOR_SPECIFICATION_VALUE, COMPETITOR_ASPECT_VALUE and COMPETITOR_PRIMARY_NEED_SCORE classes are defined to store the information about benchmark on customer PRIMARY NEEDS and NEEDS as well as SPECIFICATION and ASPECT. COMPETITOR_PRIMARY_NEED_SCORE and COMPETITOR_NEED_SCORE object classes store information related to customer needs benchmarks. As for technical assessment benchmark, COMPETITOR_ASPECT_VALUE and COMPETITOR_SPECIFICATION_VALUE classes are defined, to store this information. Their class attributes will be in the form of values assign to them. In this structure, the enterprise product is consider as one of the competitor. Table 4-4 shows these classes and thus completes the structure of the QFD information system architecture.
4.2.3 Implementation of Data Structure

The full data definition in the implementation is large and has therefore been split across four schemas purely for efficiency. Data definitions for PRODUCT, NEEDS, ASPECT and SPECIFICATION are structured together in needs_spec.ddl. Classes related to interrelationship and correlation matrices are to found in inter_correlate.ddl. The classes associated with competitors’ benchmark are grouped together in benchmark.ddl. Classes associated with PRIMARY_NEEDS class are available in primary_needs.ddl. Elements of these classes has been described by table 4-2, table 4-3 and table 4-4. Explanation about PRIMARY_NEEDS class implemented as a separate database schema can be found in chapter 7.

Achieving a working version of the software has required substantial amounts of software development, both for the data definitions and for associated implementations.
of class methods. The class methods have been written to support applications populating and modifying the Product Model database.

4.3 APPLICATION AND EVALUATION OF THE QFD INFORMATION MODEL

Syan (1994) has hinted that QFD implementation is currently manual or is aided via a static documentation process by a computer system. This has greatly restricted the potential of QFD as defects in design parameters from Computer Aided Design systems or databases cannot be automatically detected. Hence utilising the QFD information model described, will indeed provide an effective solution to the problems raised as established in chapter 2. This prototype QFD information model enables information about QFD processes be captured and stored in a database as part of the Product Model, and enables them to be assessed when required or requested. Information structure flexibility that accommodates all relevant elements of the QFD system will enhance QFD implementation.

4.3.1 Characteristics of the QFD Information Model

The QFD information system architecture is generic, in that it can support any product design. It is for this reason that the OO structure has been adopted as it allows each project team to use its own instance of the Product Model. Different databases for each product design can be managed and stored separately as Product Models that can be retrieved as and when they are needed.

All information about the customer needs, design features and competitor benchmarks, resides in Product Models as static data and can be freely used given proper and systematic management of data. The approach that QFD techniques use to capture information about the HoQ fits the rule based paradigm well. This fit can be
exploited by implementing knowledge-based software experts to develop and analyze the QFD information, as exemplified in chapter 5.

4.3.2 QFD Information Model Assessment

The use of an object database system to capture instances of the classes has proved to be very successful and powerful [Booch, 1994; Bird, 1992]. Application of an OO database to establish a QFD information structure enables information to be captured as attributes to objects within the Product Model. For example in table 4-1, NEEDS class has an object whose property description 'Easy to carry' and 'Easy to hold' have relative importance of 5 and 4 respectively. The flexibility of capturing this information in the product model makes it available to all designers throughout the design process.

This approach of structuring QFD information has enhanced the process of capturing information as established by previous literature. Bird (1992) established the use of an expert system to capture QFD information and defined the state of an object to be a segment of the system that aggregates various processing and inferential elements such as rules and procedures and a class to be a frame that described the structure and content of data attributes. Figure 4-2 illustrates a section of the rule described by Bird.

---

**Figure 4-2** Type Hierarchy and Inheritance for Class CUSTOMER_NEEDS and Sample Rule Set for Frame Generation and Assignment of Slot Values [Bird, 1992]
Whilst this approach provides a reasonable support for small systems, its efficiency to capture information for large systems is suspect as many rules have to be coded to define an individual object within the system.

Another approach was proposed by Sriraman et al (1990). They established a hierarchical relationship of objects to structure QFD information in the form of parent object at its highest level and instances of the objects that inherit information from their parent. Sriraman's approach used voice of customer (VOC) as its parent and through this, objects representing the systems were built that inherit from the VOC at second level. The third level contains objects representing parts; specifications for the parts are represented at the fourth level; the fifth and final level is represented by several processes that are required by product specifications. As the proposed structure focused on hierarchy of objects and frames as depicted in figure 4-3, similar problems are anticipated if the structure is applied to large systems as the hierarchy of information grows.

<table>
<thead>
<tr>
<th>V.O.C.</th>
<th>fuel pump</th>
<th>carburetor</th>
<th>fuel line</th>
<th>fuel tank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long Lasting</td>
<td>springs</td>
<td>diaphragm</td>
<td>cams</td>
<td>rocker</td>
</tr>
<tr>
<td>(durability and reliability)</td>
<td>fuel pump</td>
<td>springs</td>
<td>fatigue</td>
<td>no. of coils</td>
</tr>
<tr>
<td></td>
<td>fatigue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fatigue life</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shot peening</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 4-3 QFD Matrix for a Fuel Pump [Sriraman, 1990]*
The author strongly believes that the process of capturing QFD information is not restricted to gathering important elements in the QFD process, but must include provision of the information needed by designers to guide them when defining design parameters. Hence, the process of capturing information must be able to relate customer needs and design features, as well as competitor benchmarks. The flexibility of model structure to include this process of capturing information proves to be vital in the QFD process. The proposed structure provides this functional feature which has not been addressed by information models proposed by either Bird or Sriraman.

4.4 SPECIFICATION OF THE QFD INFORMATION IN PRODUCT MODEL

This section describes features that act as a supporting tools during the capture of QFD information. This is followed by descriptions of weighting scales used to indicate the strength of attributes applied by elements of the HoQ. The examples exhibited by display windows were taken from a mug design case study described in chapter 6.

4.4.1 Graphical User Interface

The research has not been focused on developing a sophisticated graphical user interface. Hence it has used available tools as far as possible to interface with the application software. These tools are described here so that they can be used hereafter without tedious explanation.

(a) Window Utility

Invoking the software in executable file 'house_of_quality' causes the active window shown in figure 4-4 to appear on the screen. This window has two menu
buttons, *File* and *Application*. The *File* menu begins populating QFD information, whilst the *Application* menu enables the process of populating the KRM rules.

The window utility is implemented using Motif Widgets compiled in window class. This window utility allows integration with the data stores in the database as depicted by figure 4-5. The example shown in this figure illustrates the list of ASPECT objects retrieved from the database ready for selection. The window utility is event-driven and only activates after an event is selected. For instance, if the designer selects the object 'a1', the object1 name will appear in the selection box and a click on the 'OK' button will register the event. The selected argument that appears in the selection box will be passed to the next function for further processing.

---

1 The object name is used to identify the unique object for selection. However to help the designer recognise the required object, a description of the object is specified in brackets.
(b) Database Tool Manager

The Database tool is a utility program provided by Objectivity/DB for database development and maintenance. The most commonly used database tool is the database tool manager. It is a graphical application that opens a Federated Database Object (already discussed in chapter 3) and launches Objectivity/DB database tools that operate on the Federated Database Object. This application incorporates a graphical environment that fully conforms to Motif user interface guidelines. Figure 4-6 illustrates an application that uses the tool manager browser to view the content of the database. The selection of the browser will open the Federated database and within the Federated database three other types of database objects are opened.

![Figure 4-6 A Typical Objectivity/DB Tool Manager Browser](image)

Database, Container and Basic objects (described in chapter 3) store captured QFD information. Selecting the Database objects enables the manager to
browse through the Basic objects available in the database as shown by figure 4-7. In this figure, when CUSTOMER_NEEDSprod_1 is selected, the manager will list all data of the Basic objects available for this Database object. As the example shows, a Basic object 'sn2-2' is selected, so the attributes attached to this Basic object are listed. Hence, the Tool Manager gives the designer an opportunity to browse through all information that has been captured and to identify any data that needs specific attention or modification. Modification is done through the application software.

Figure 4-7 A Typical Example of Attributes Listed in Basic Object

4.4.2 Getting the Correct Facts into HoQ

The process of capturing QFD information and using the HoQ is perceived as simple. Successful implementation of the HoQ involves the collection of the voice of
the customer data, evaluating the collected data and translating these subjective data into specifications. The task involves applying numerical scales to indicate the priority of data collected. This section discusses the descriptors derived for each numerical scale used by specific parameters to determine their priorities. The empirical values given to describe each descriptor are not absolute values and are obtained from relevant literature [Ginder, 1991; Shillito, 1994]. Designers can use other scales to indicate the importance of each priority.

(a) Quantify Customer Needs Importance

The first subjective evaluation found in the HoQ is to record customer needs importance according to a weighting scale as shown by table 4-5. These priorities help to allocate engineering resources and guide the team when they are forced to make trade-offs among customer needs [Griffin et al, 1992].

(b) Quantify Relationship between Customer Needs and Design Features and between Design Features

The Relationship matrix is established to relate the strength of relationship between customer need and design feature. As described in chapter 2, the software allows the user to select one out of four (Strong, Medium, Weak and None) values. Once the relationship strength is selected a numerical value is assigned by the software to indicate the strength of each of the selected relationship as depicted by table 4-6 (a). The values are relative weighting so that strong relationship are valued heavily while relationships that are non-existent or weak carry little value.

Similarly, evaluation between two design features resulted in establishing a correlation matrix. The software offers two types of selection to indicate the strength of its correlation. If the correlation is between the same design feature, the system only
permits one selection of NONE. Again each relationship is assigned a numerical value.
Table 4-6 (b) shows the numerical values assigned in the correlation matrix when the
correlation among design features is examined. The correlation provides a range of
recognition from strongly positive and strongly negative correlated design features as
determined by the design team. Both positive and negative correlations provide
important information. Positive correlation help identifies design features that are
closely related and negative correlation represent conditions that will probably require
trade-offs.

(c) Quantify Competitive Assessment

Competitive assessment is used to gauge the current position of an enterprise with its competitors. This is usually accomplished by comparing the customer
evaluations of the competition compared to in-house evaluations of current competitive standing. While capturing competitive benchmarking information the system records
values obtained by each competitor with respect to every customer need. This scale
value of 1 to 5 describes the performance of each competitor. Descriptors of each scale
value are defined in Table 4-7.

Design feature benchmarking (also known as Technical Benchmarking) is
recorded in the same way as benchmarking satisfaction of customer needs.

(d) Quantify Difficulty

Each Design Feature is rated for the degree of technical difficulty of changing or maintaining its target value in the design product. A possible scale of 1 to 5 (depicted by table 4-8) can be used to define difficulty, providing the basis for a plan of action to improve the appropriate design features [Bicknell et al, 1995].
### Chapter 4

#### Table 4-5 Weighting Scale for Customer Needs Importance [Ginder, 1991]

<table>
<thead>
<tr>
<th>Score</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Critical, would impact the buying decision</td>
</tr>
<tr>
<td>4</td>
<td>Very important, impacts the buying decision when considered with other demands</td>
</tr>
<tr>
<td>3</td>
<td>Like to have, would spend money for it</td>
</tr>
<tr>
<td>2</td>
<td>It would be nice, would like to have, but would not spend money for it</td>
</tr>
<tr>
<td>1</td>
<td>I don’t care</td>
</tr>
</tbody>
</table>

#### Table 4-6 (a) Numerical Weighting for Relationship Matrix  
(b) Numerical Weighting for Correlation Matrix

<table>
<thead>
<tr>
<th>Score</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Strong Positive</td>
</tr>
<tr>
<td>3</td>
<td>Positive</td>
</tr>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>-3</td>
<td>Negative</td>
</tr>
<tr>
<td>-9</td>
<td>Strong Negative</td>
</tr>
</tbody>
</table>

#### Table 4-7 Benchmark Scale for Competitive Assessment [Ginder, 1991]

<table>
<thead>
<tr>
<th>Score</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>World class: Meets the expectations beyond the customer’s wildest dreams</td>
</tr>
<tr>
<td>4</td>
<td>Best in the class: There aren’t any better</td>
</tr>
<tr>
<td>3</td>
<td>Average: Pretty good but there is room for improvement</td>
</tr>
<tr>
<td>2</td>
<td>Disappointing: You should have been better</td>
</tr>
<tr>
<td>1</td>
<td>You are in a class by yourself! There is no one as bad as you are</td>
</tr>
</tbody>
</table>

#### Table 4-8 The Difficulty Rating Scale [Shillito, 1994]

<table>
<thead>
<tr>
<th>Score</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Major difficulty; invention required; unknown technology.</td>
</tr>
<tr>
<td>4</td>
<td>Difficult; technology not immediately available; major development work required</td>
</tr>
<tr>
<td>3</td>
<td>Some effort needed; no invention required; some development work needed</td>
</tr>
<tr>
<td>2</td>
<td>Little effort; technology known; some development needed but not major</td>
</tr>
<tr>
<td>1</td>
<td>Easy; known technology; off-the-self; plug-in</td>
</tr>
</tbody>
</table>

... Development of QFD Information Model
4.4.3 Procedures to Capture QFD Information

The beginning point for processing through the QFD system is, in accordance with the QFD technique, acquiring information about the product and creating the HoQ. Figure 4-8 illustrates a simple flow diagram that is used to describe steps involved in capturing and analysing QFD information. Steps 1 to 3 involve populating the QFD information while step 4 uses KRM rules to analyse the matrix and prioritise design attributes. The implementation of step 4 leading to populating of rules is described in chapter 5. Two case study examples have already been implemented and details of their implementations have been described in chapter 6 as they represent two different cases that provide significant validation of the software.

4.4.3.1 Populating Customer Needs and Competitors’ Information

The structure of the QFD information requires an instance of the PRODUCT class to be populated first. This is achieved by selecting the create product sub-menu found in file menu. The system asks for the object name and its description. It is recommended that the object name for the product be as short as possible because the product name will be linked to subsequent database object created that belong to this particular product as shown in figure 4-7. prod_1 is a PRODUCT object name linked to database objects uniquely belong to the product.
Chapter 4

Figure 4-8 A Simple Flow Diagram of a QFD Process
Chapter 4

After creating the PRODUCT object, the next task is to create PRIMARY_NEEDS objects which are instances of the PRIMARY_NEEDS class, by selecting the Create Primary Need sub-menu. The system requires the selection of product name from the list displayed as shown by figure 4-9.

![Figure 4-9 The Selection Box for Product Object Name](image)

The selection enables PRIMARY_NEEDSprod_1 database to be created and in it required Basic objects are created that store instances of the PRIMARY_NEEDS class with attributes as described by table 4-2. Similarly, customer secondary needs are then created using the same approach. Selecting the create_customer_need sub-menu enables instances of the NEEDS class to be created and stored in CUSTOMER_NEEDSprod_1 database object. A new user interface function (as depicted by figure 4-10) is introduced to guide the user in capturing customer secondary needs information. The function enables the user to choose creating secondary needs continuously from each primary need, or creating from selected primary needs or creating the secondary need individually and later relates the created secondary needs with particular primary needs.
As an example, selecting option one enables a list of primary needs to appear on the screen as shown by figure 4-11 and by selecting one primary need from the list enables the system to create secondary needs object associated with the selected primary needs. This process continues until the user indicate the system to stop by sending 'no' message to the prompt dialogue box.

Information about competitive evaluation are captured next. Before this information is captured, the system requires the user to create instances of COMPETITORS class. These COMPETITORS class objects are captured in a
COMPETITORSprod_1 Database object. Once the COMPETITORS object has been created, customer need benchmarking values for each competitor are captured.

### 4.4.3.2 Populating Design Features

The process of capturing design feature information is divided into two stages and information is captured using ASPECT and SPECIFICATION classes. This process uses similar methods of capturing the information described above to capture primary and secondary needs, but using a different process module. Populating instances of ASPECT and SPECIFICATION classes are done by activating create_aspect and create_specification sub-menu respectively, found in the file menu. Instances of ASPECTs must be populated first before instances of SPECIFICATION can be created. Invoking create_aspect and create_specification will create ASPECTSprod_1 and SPECIFICATIONSprod_1 database objects respectively.

### 4.4.3.3 Populating Technical Competitive Benchmark Values

Technical Competitive Benchmark values are populated using create_competitive_value sub-menu. This will create the BENCHMARK_SPECVALUEprod_1 database object, thus enabling instances of the COMPETITOR_SPECVALUE class to be created. The populating process uses a similar method that used to populate competitive benchmark for customer needs described in previous section.

### 4.4.3.4 Populating Interrelationship Matrix

The Create_specInterrelation sub-menu is selected which open CUSTOMER_NEEDSprod_1 and SPECIFITIONSprod_1 database objects to enable
connection of the interrelationship matrix once created. This will create SPEC_INTERprod_1 database object thus allowing the instances of the SPEC_INTERRELATION class to be captured and stored in the database.

4.4.3.5 Populating Correlation Matrix

The Correlation matrix (also known as roof of the HoQ) is populated by invoking the create_spec_correlation sub-menu found in file menu. This creates the SPEC_CORRprod_1 database object and enable instances of SPEC_CORR class to be captured and stored.

4.5 SUMMARY

This chapter presents a new approach to the functional decomposition and modelling of QFD information. It provides the basis of a modelling and analysis approach based on OO design methodology which places great emphasis on the concepts of encapsulation, abstraction and modularity. The importance of the approach lies in its inherent ability to capture and store information that is interrelated, in a well structured manner. A significance of the model is in its contribution towards the understanding of processing QFD information step by step and at each step maintaining the relationships within the information.

Theoretically, the QFD information structure so designed is promising in terms of its properties of portability and promoting software reuse. Its generic structures to capture HoQ information has been implemented in demonstrator prototype software. Its value is assessed through experimentation to be described in chapter 6.
5.1 INTRODUCTION

As described in Chapter 4, the author has proposed the QFD information model architecture to capture and store QFD information in the Product Model. This chapter demonstrates how the QFD information can be exploited once it is in the Product Model.

A QFD software expert development which adopted an established KRM methodology developed in previous research is discussed. The QFD software expert will become a key resource tool that provides the support to analyse information captured by the QFD information model. Its ability to utilise the captured QFD information within the Product Model systematically will enhanced the described system. This chapter therefore establishes the QFD software expert architecture that will contribute to the development of the QFD decision support system (QFD DSS).

The main objective of this QFD DSS is to define design feature priority so as to provide support for designers by identifying critical design features. Hence, this chapter introduces a software expert structure that will facilitates implementation of an algorithm devised to achieve the objective. A mathematical model is developed as part of the algorithm. The whole system, which is embedded in a single federated OO database along with the Product Model (to minimise computing resource needs when
interacting with other information models) is explained. A simple example to demonstrate the concept is discussed prior the discussion on the implementation of the algorithm.

A substantial amount of software development effort has been put to get the working version of the software expert running. However, the author acknowledge that the system is not without flaws. There are still improvements that can be enhanced to made the software more suitable for commercial consumption. As the priority of this research is to demonstrate the ability of the structures to support QFD information acquisition process, effort on these areas have become the author’s secondary priority. These include, developing a more elegant user interface capability and using a more robust database system that could provide interactive data definition and manipulation language such Structured Query Language (SQL).

5.2 DEVELOPMENT AND REALISATION OF KNOWLEDGE BASED REPRESENTATION MODEL

5.2.1 Background

The use of AI in design and manufacturing activities has been slowly increasing since its introduction more than twenty years ago [Bird, 1992]. There is extensive literature on AI, and particularly on Knowledge Based Systems [Popplewell et al, 1995; Blount et al, 1994; Bird, 1992; Oliver, 1990]. Knowledge Based Systems are computer systems that are programmed to include an internal representation of know-how about a particular task [Blount, 1994]. This know-how is used to solve problems, give advice and draw inferences. It is becoming a necessity to incorporate a knowledge base system in design and manufacturing activities because many of the design and manufacturing problems are too complex for traditional optimisation methods [Bird, 1992].
QFD is no exception. Whilst the technique is simple to use and its results can be
displayed in a graphical model that is easy to comprehend, a significant degree of
difficulty arises as the magnitude of the resulting HoQ matrix increases [Zairi, 1993].
The use of manual methods becomes insufficient. It is generally believed [Syan, 1994;
Cohen, 1995] that the implementation of a knowledge-based expert system to manage
QFD information is essential in order to support QFD application.

The QFD information model structure described in chapter 4, captures the whole
logical structure of QFD information in Product Model. This enables the users to
manipulate the information systematically. However, in implementing QFD, the task of
analysing the captured information must not be underestimated. The analysis process
could be simple but if proper supporting tools are not available it becomes a laborious
task, and this has discouraged many potential users. It is indeed true that to be able to
utilise all the eight elements of the HoQ during the analysis process and visualise them
in a coherent environment is very demanding. Furthermore, if the magnitude of the
matrix becomes very large, it is troublesome to provide an effective solution. Hence, the
proposed decision support system, a QFD Software Expert is intended to provide
solutions to these problems.

The QFD software expert is designed and built as a knowledge based expert and
resides as elements of a single federated OO database along with the Product and
Manufacturing Models to minimise the overhead of computing resources when
interacting with other information models. In order to facilitate interactions with the
database structures the application is written in C++. The object structure of the
software expert is built as an instance of the KRM [Harding, 1996b].

5.2.2 The Knowledge Representation Model Concept

A detailed description of KRM concept can be found in Harding (1996b). Its
structure is summarised here as it will be used to develop the QFD Expert System. The
KRM defines the structures of software experts and captures them in the federated OO database together with Product and Manufacturing Models as illustrated by figure 5-1.

The structure of the KRM enables any software expert to be developed and built as an instance of the KRM. Hence, the QFD software expert is developed as an instance of the Software Expert class. The class structure of the software expert built as an instance of the KRM modelled using Booch’s graphical notation is shown in figure 5-2. Software Expert class object is defined from Expertise class object which may come from human experts or from software expert applications or possibly from both referred as agents [Harding, 1996b]. Each software expert class is composed of an Expert_Module class which is associated with three other classes, a Working_Memory class, an Inference Engine class and a Knowledge Base class.

Figure 5-1 The Knowledge Representation Model [Popplewell et al, 1995]

Figure 5-2 KRM Software Expert Structure [Harding, 1996b]
The system will process the knowledge through the inference engine that provides the software expert with the ability to apply and use its knowledge. This knowledge is captured by a Knowledge Base object that contains class hierarchies of Rule_Set and Rules, based on the production rule paradigm. They are represented in the form of if ... then rules that are associated with Condition and Action classes. The Condition class contains an Expression class that evaluates when the rule may be applied to a problem instance. The Action class defines the associated problem solving step. Figure 5-3 illustrates the breakdown of these classes. A number of classes, specialised from the Expression and Action classes, are developed to form part of the QFD software expert.

Figure 5-3  A Representation of Knowledge, Captured Within Rule and RuleSet Classes, Using Booch Object Oriented Design Graphical Notation [Harding, 1996b]

The Working Memory class defines a Working Memory object that stores variable information to be used in association with the expert's domain knowledge, possibly to facilitate the processing of that knowledge [Harding, 1996b]. This Working Memory class is a parent class from which a hierarchy of domain specific working memory objects including the QFD working memory are specialised.
5.3 SELECTION OF THE QFD EXPERT SYSTEM ARCHITECTURE AND ITS METHODS

5.3.1 QFD Software Expert Architecture

The QFD software expert inherits from the basic KRM class, the Expert Module described in the previous section. In order to achieve its functionality, the processing of the QFD software expert is done by the interactive behaviour of Knowledge Base, Working Memory and Inference Engine objects. Figure 5-4 illustrates the class structure of the QFD software expert.

![Diagram of QFD software expert class structure](image)

**Figure 5-4** The Class Structure of the QFD Software Expert Using Booch Object Oriented Graphical Notation

5.3.2 Working Memory Class for QFD Software Expert

The QFD_W_M class is specialised inherits from the Working Memory class to facilitate processing of QFD knowledge. This QFD_W_M stores temporary information that is originated from QFD information model. The Inference Engine evaluates the QFD Knowledge Base through the production system metaphor and update values in the Working Memory. After the first production rule is fired the control cycle repeats with the modified Working Memory value using the next production rule. The process terminates when there are no more rule conditions true.
5.3.3 Knowledge Base Class for QFD Software Expert

A Knowledge Base object contains knowledge of how the QFD software expert can collect, update and analyse QFD information values within the Product Model. The KRM approach has enabled the expert knowledge of the QFD software expert to be captured, using a production rule approach at its highest level, but can embrace other knowledge paradigms, including for example, neural networks, genetic algorithms or Fuzzy Sets Theory [Fung, 1997] if required. The approach used enables the knowledge to be stored within an OO database, and has the capability of capturing a wide range of behaviour.

The QFD software expert utilises the inheritance structures and polymorphism capability of OO programming to enhance this Knowledge Base class. Knowledge associated with any particular Knowledge Base object, is stored through the definition of Rule_Set and Rule objects. Key classes in these Rules classes, Expression and Action classes are involved in the processing of capturing knowledge. They are parent classes for a wide variety of objects which are similar to the extent that they all demonstrate a specific type or aspect of behaviour. Enhancements of the established structures are achieved by developing classes that inherit from both Expression and Action classes as depicted by figure 5-5 and 5-6.

![Figure 5-5 A Representation of the Expression Class, of the QFD Software Expert, Using Booch Object Oriented Notation](image-url)
5.3.4 Implementation of Data Structure for QFD Software Expert

Implementation of the QFD software expert uses a similar approach to that used to implement the QFD information structure: defining the class structures in OO database schemas. Modularity capabilities available in OO programming enable the full data definition to be split into three manageable schemas and has enabled the software expertise to be captured in the Objectivity/DB (an OO database system described in chapter 3).

Classes associated with the KRM classes are coupled together with the general purpose classes, and inheritance of the Expression and Action classes in the rules.ddl database schema. New structures developed as part of the QFD software expert that inherit from the Expression and Action class can be found in mod_rules.ddl and
mod1_rules.ddl. QFD Working Memory class that is associated with the KRM Working Memory are structured together in working_memory.ddl. Details of these data definition schemas are to be found in appendix I of this thesis. List of rules are also to be found in appendix II.

5.4 THE IMPLEMENTATION OF THE KNOWLEDGE REPRESENTATION MODEL

The procedures below demonstrate the flexibility of manipulating captured information stored as a persistent database, realised by using the KRM rules.

5.4.1 Populating the Knowledge Representation Model

The KRM is populated as a database object containing all the necessary Rule_Set objects and their associated objects. These are the general rules normally called HOQ_RULES, comprising the knowledge about how design attribute priorities are implemented. Their implementation is realised by selecting create_HoQ_rules sub-menu found in Application menu. This process enables instances of Rule class and its associates to be captured and stored in the HOQ_RULES database.

A Rule class is created by selecting the create_rule sub-menu found in the Application menu. The system will ask the user to identify a specific product from a list (see figure 4-9) that will be associated with the rules to be created. Following the selection, a window depicted by figure 5-7 (a) is displayed on the screen. The user needs to select the second option to enable instances of Rule class to be created.

The system begins by creating the Rule_Set object and capturing its attributes that is its description and Rule_Set number (this number will enable the expert to process the
rule according the sequence of this number in ascending order). The software will then allow the user to create Rule objects and its associates. The first rule’s associate is the rule’s condition defined by selecting a simple condition or compound condition using the selection window depicted in figure 5-7 (b). A compound condition is a set of simple conditions.

After selecting the condition type that the rule is associated with, the software requires the user to select type of display mode that will be used to display messages that will guide the user through the process of using the QFD expert. Figure 5-8 shows the list of options that the message can be displayed. The term \( WI \) indicates that the display mode will use motif window to display the message. The system then requires the user to type the message to be displayed. Questions such as shown by figure 5-9, are captured and will be displayed while processing the KRM rules providing the guidance necessary for the user to accomplish the task.
Rule associated with the resulting action is defined next. Resulting action can also be a simple resulting action or a compound resulting action (executed by sequentially performing the actions of its associated simple actions and another action [Harding, 1996b]). This is accomplish by selecting the option from the displayed selection window as depicted by figure 5-10 (a). The behaviour of an object of the Simple Resulting Action class includes the ability to execute a set of instructions or duties [Harding, 1996b] as depicted by figure 5-10 (b). The selection of the option from the selection window will create an instance of the selected Knowledge object class that is used to formulate specific rule.
5.4.2 Algorithm to Determine Mean Value for Interrelationship Weighting Scale

The aim of this first example is to demonstrate the concept of populating an instance of the KRM. It is a 'simple' expert to assist a designer in converting specification interrelationship weighting scale value into aspect interrelationship weighting scale value. The expert assumes at the beginning, the Product Model already contains specification interrelationship weighting scale values. After processing the knowledge, the Product Model will contain aspect interrelationship weighting scale value.

Conceptually, the 'ASPECT MEAN VALUE EXPERT' has initially been modelled with a Working Memory and a Knowledge Base object. The implementation of the KRM is best described by the answering the following questions.

*Where methods come from?*

*What does the expert have to do?*

*What information does the expert need to work efficiently?*

**Implementation:**

*Where methods come from?*

\[\text{Figure 5-11 The Implementation of Rule Class for the Expert}\]
The implementation of the Expert utilised the WI_COMPUTE_ASPECT_MEAN_ACTION class (see figure 5-11) which acquires knowledge for the Expert in the form of rule:

**If** a process requires to determine the weighting scale value of ASPECTs, **then** compute SPECIFICATION mean value.

Figure 5-12 and figure 5-13 exhibit the modelling of Knowledge Base and Working Memory respectively. Using this approach, once the need to determine the specification interrelationship weighting scale mean value is established, the Inference Engine (see section 5.2.2) will process the knowledge and updates aspect interrelation weighting value in the Product Model.

*Modelling of Knowledge Base:*

*What does the Expert have to do?*

*Figure 5-12 Knowledge Base Object*
Modelling of Working Memory:

What information does the Expert need to work efficiently?

5.4.3 Algorithm to Determine Design Feature Priority for QFD Software Expert

In the second example, information about the HoQ is used by the QFD Software Expert to define the knowledge content to provide advice to support designer determine specifications that require modification according to its priority. The established algorithm assists the designer to identify which specifications need to be targeted for design changes. Through the use of an underlying QFD information system, specific data from the database is managed and processed to determine coefficient values. In this way the relationship of customer needs and specifications defined by the interrelationship matrix and correlation matrix stored by the QFD information system are utilised systematically without any difficulties. Once the rules are captured as instances of the KRM, the knowledge content should have the following requirements:

- It has the ability to check the status of NEEDS object of own product as compared with competitors benchmark. Set_Init_Benefit_Values class object provide this function that enables the expert to validate own product’s need score value. If the value is greater than competitors’ need
If the ability to look at the interrelationship object and compare the specification value of own product with competitors' benchmarks. The Check_Benefit_Value_Spec_Benchmark object provides this function by comparing own product's specification value with competitors' specification value; if it is greater, then this column of specification for the NEEDS object will be given a value of zero.

Table 5.1 Knowledge That Check and Updates Each Coefficient Value if Need Score for Own Product is Greater than Competitor Need Score

<table>
<thead>
<tr>
<th>Specification</th>
<th>Need Score</th>
<th>Coefficient Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>n capacity</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n total mass</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n thermal</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n angle</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n surface</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n mass</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n size</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n thickness</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n hardness</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n texture</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Knowledge That Check and Updates Each Coefficient Value if Need Score for Own Product is Greater than Competitor Need Score
It has the ability to correlate Interrelation and the Correlation Specification objects that affect specifications under observation. Class object Check_Benefit_Values_Relationship evaluates this process by using Equation 1 to define the relationship of the interrelationship and correlation matrices.

\[ v_i = \{ \sum f_c(\xi_j)G(I_j) \} + G(I_i) \] \hspace{1cm} \text{Equation 1}

where: 
- \( v_i \) coefficient value of the specification under observation
- \( f_c(\xi_j) \) correlation value of the knock on specification
- \( G(I_j) \) interrelation value of the knock on specification
- \( G(I_i) \) interrelation value of the specification under observation

| Table 5-2 Knowledge that Check and Updates Each Coefficient Value if Specification Score for Own Product is Greater than Competitor Specification Score |

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe to use</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimal heat transfer</td>
<td>0</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
It has the ability to total coefficient values for customer needs for every specification of each column.

It has the ability to sort coefficient values after these values have been totalled and notifies designer which specifications (with respect to the coefficient values) require modification in order of priority.

5.5 SUMMARY

This chapter has highlighted the development of a QFD software expert. It has adopted a KRM concept developed by previous research. The class structures of the KRM has been enhanced to include classes that enable the possibility of manipulating QFD information within a Product Model. The QFD software expert also uses the OO design methodology to capitalise on its potential features such as inheritance, polymorphism and modularity.

The additional Knowledge Base classes developed as part of the KRM provide the support that is required by designers and QFD implementers to explore the potential of the QFD technique without with the effort seems to be without ends because of the magnitude of the HoQ matrix. Whilst the algorithm presented here for determining target values is almost certainly capable of improvement, it exemplifies the potential for systematic interpretation of QFD information once this is captured and accessible within the Product Model. The introduction of the mathematical model into the software expert structures has led to the understanding that other intelligent agents can be integrated into the system to manipulate the captured QFD information.

Besides successfully developing the QFD software expert, it is also necessary to provide a reasonable experimentation problems that can validates the developed software expert. This forms the main emphasis of the next chapter.
6.1 INTRODUCTION

The performance of the QFD information model needs to be evaluated in respect of its capability to support product design and promote CE activities. The procedure for capturing QFD information and analysing the captured information by using the designed software is also presented. Two case studies have been described to demonstrate the performance of the software, leading to solution of the problems raised in chapter 2. The first case study described a simple approach towards understanding the implementation of the concept that is used in the design of the software. An industrially based problem is exemplified by a second case study to validate the information structure's potential in designing a product in a real engineering design environment.

6.2 CASE STUDY 1: A MUG DESIGN

The first case study used to demonstrates the value of the software is to capture HoQ information about a mug. The author acknowledges that the information collected may be immature in a certain aspects, but it is adequate to demonstrate the software
capabilities in capturing and analysing information about the product. This example of a mug design is used because of its being a consumer product that is commonly used and easy to comprehend.

It is important to establish that this research does not focus on procedures for gathering customer needs and benchmark information, hence standard procedures as established in existing literature [Ulrich et al, 1995; Griffin et al, 1992] are adopted. The approach used to capture and analyse the information, as in figure 4-8, is described as follows.

6.2.1 Customer Needs, Relative Importance and Competitive Evaluation

Customer needs are collected by interviewing office staff and the outcome of the interviews revealed eight significant needs that are taken for the experiment. These customer needs were grouped and clustered together based on similar attributes and further aggregated to form a two level hierarchy as shown shaded on the left of table 6-1. These aggregated list typically consist of a set of general primary needs, each one of which is further characterised by a set of secondary needs expressed in more detail.

Customers were also asked about their preference with regard to the needs and numerical importance weighting (as shown by table 4-5) is given to each need. This numerical value is essential to make trade-offs and allocate resources in designing the product. Benchmark evaluations with respect to each customer need for the mug are obtained from two different competitors. These evaluations together with the evaluation from enterprise's own product is information captured as customer competitive benchmark. Each evaluation is assigned a value using weighting scale from table 4-7. The values are shown on the right hand side of table 6-1.
### Analyse Competitive Benchmarking

### Customer Needs (whats)

#### Table 6-1 List of Customer Needs, Customer Relative Importance and Competitors' Benchmark

<table>
<thead>
<tr>
<th>Customer Needs</th>
<th>Customer Relative Importance</th>
<th>Competitor Benchmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price in value</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Ease of ordering</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Other features</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total spec score</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Assessment of QFD Information Model**
- **Chapter 6**
6.2.2 Development the Technical Portion of the HoQ Matrix

There are four important steps that need to be addressed in this section. These include populating design features, design feature competitive benchmarks, interrelationship matrix and correlation matrix.

6.2.2.1 Populating Design Features

Before populating instances of aspect and specification it is advisable to contemplate each customer need and to consider what precise, measurable characteristic of the product will reflect the degree to which the product satisfies that need [Ulrich et al, 1995]. The designer should be able to come up with a list of aspects and its detail specifications. Table 6-2 shows a typical example of aspects and specifications of a mug recorded horizontally at the top of the customer needs. Note that an aspect may also be the specification of the product as shown in table 6-2. An aspect Surface Decoration (SD) is also a Surface Decoration of the specification recorded in the last column of the design feature.

6.2.2.2 Populating Technical Competitive Benchmarks Values

Table 6-3 listed the technical competitive benchmark values that are captured using this process.

6.2.2.3 Populating Interrelationship Matrix

The process of populating the interrelationship matrix is depicted by table 6-4. In this example, the interrelation between secondary customer needs and specification were populated.
6.2.2.4 Populating Correlation Matrix

Table 6-5 illustrates information about the roof that is used to examine the correlation between design features.

6.2.2.5 Display QFD Information

QFD information captured can be displayed and viewed. This is done by selecting *Display_the_House_of_Quality* sub-menu found in the *file* menu. A window will display the HoQ with the captured information. The design of this display window allows the designer to scroll the display window if the application is so large that does not fit the display monitor. Another important feature available in this display window is the display orientation of the roof of the HoQ which is slightly different from the conventional roof of the HoQ, but maintaining its original concept.

Detail of the display window is depicted by figure 6-1 (a) and (b). In figure 6-1 (a), shows a portion of the display window that illustrates information about primary needs, secondary needs, design aspects, specifications, interrelationship matrix weighting scale values and technical assessment benchmark values, whilst figure 6-1 (b) shows information about needs benchmark values.
### Table 6.2 List of Aspects and Specifications that Contemplate Customer Needs for Mug Design

<table>
<thead>
<tr>
<th>Customer Needs</th>
<th>Specification</th>
<th>Container</th>
<th>Handle</th>
<th>Molding</th>
<th>Surface Treatment</th>
<th>Heat Resistant</th>
<th>User Outlook</th>
<th>Consumer A</th>
<th>Consumer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe Design</td>
<td>Safe to use</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Minimal heat transfer</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value for Money</td>
<td>Easy to carry</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Easy to hold</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Visually attractive</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reasonable price</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Does not collect dirt on surface</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Easy to clean</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**TOTAL SPEC. SCORE**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Consumer A</th>
<th>Consumer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Value for Money</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Performance</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**NORMALIZED SPECIFICATION RATING**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>1</td>
</tr>
<tr>
<td>Value for Money</td>
<td>3</td>
</tr>
<tr>
<td>Performance</td>
<td>5</td>
</tr>
</tbody>
</table>

**QFD 4 Matrix**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Container</th>
<th>Handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Value for Money</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Performance</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
**Technical Competitive Benchmark**

<table>
<thead>
<tr>
<th>SPECIFICATION</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUFACTURING</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Volume</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Material's Thermal Conductivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bending Power Angle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface Texture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. Weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross sectional thickness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Material's thermal Conductivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part's Strength Hardness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface Texture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit Direct Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strength of joint</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface deformation (SD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 6-3 The Technical Competitive Benchmark Values**

<table>
<thead>
<tr>
<th>Spec Value own product</th>
<th>Spec Value competitor A</th>
<th>Spec Value competitor B</th>
<th>Target Spec own product</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 (4) 0.3 (5) (3)</td>
<td>2 (3) 0.3 (5) (3)</td>
<td>15 (3) 0.15 (3) (3)</td>
<td>15 (3) 0.15 (3) (3)</td>
</tr>
<tr>
<td>1.2 (3) (5)</td>
<td>3 (4) 3.2 (4) (4)</td>
<td>4 (5) (3) 3.2 (4) (5)</td>
<td>4 (5) (3) 3.2 (4) (5)</td>
</tr>
<tr>
<td>120 (1) 1.7 (2) (2)</td>
<td>80 (5) 3.6 (4) (4)</td>
<td>100 (2) 2.5 (5) (5)</td>
<td>100 (2) 2.5 (5) (5)</td>
</tr>
<tr>
<td>3.3 (5) (3)</td>
<td>8.5 (5) (2) (3)</td>
<td>2.5 (5) (3) 1.8 (5)</td>
<td>2.5 (4) (5) 1.8 (5)</td>
</tr>
<tr>
<td>2.1 (3) (5)</td>
<td>10 (4) (3) (4)</td>
<td>3.2 (3) 4.0 (1) (1)</td>
<td>3.2 (3) 4.0 (1) (1)</td>
</tr>
<tr>
<td>5 (5) 2.4 (5) (5)</td>
<td>4.0 (1) (1)</td>
<td>5 (1) 7.0 (4) (1)</td>
<td>5 (1) 7.0 (4) (1)</td>
</tr>
</tbody>
</table>

... Assessment of QFD Information Model
Information is evaluated by asking how each Design Feature meets the Customers' Needs.

### Table 6-4 Correlate Design Features to Customer Secondary Needs

<table>
<thead>
<tr>
<th></th>
<th>Container</th>
<th>Handle</th>
<th>M. Utility</th>
<th>C. Utility</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe Design</td>
<td>Safe to use</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Value for Money</td>
<td>Easy to carry</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Performance feature</td>
<td>Easy to hold</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Performance feature</td>
<td>Visually attractive</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Performance feature</td>
<td>Reasonable price</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Performance feature</td>
<td>Does not collect dirt on surface</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Performance feature</td>
<td>Easy to clean</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 6-4 Correlate Design Features to Customer Secondary Needs*
Chapter 6

Identify Technical Interactions

Table 6-5 Mug Design - Identified Correlation Matrix
Figure 6.1 (a) A Portion of the House of Quality Showing the Captured Information about a

...Assessment of The QFD Information Model
Figure 6-1 (b) The Other Half of the House of Quality Showing the Captured Information about a Mug

Assessment of The QFD Information Model

106
6.2.3 Analysis of the HoQ and Prioritising Design Attributes

Before the HoQ can be analysed and design features' priorities defined, all elements of the HoQ must be determined.

6.2.3.1 Populating the KRM

KRM rules used to describe how design specifications' priorities are listed in table 6-6. Whilst processing each rule, a message asks whether the user wishes to continue executing the rules. If the response yes, the system will execute the resulting action and continue to process next rule. If no, then the system will continue to process next the rule without executing the resulting action of that particular rule.

6.2.3.2 Processing the KRM

KRM rules are invoked using the Run Application sub-menu found in the Application menu. The Run Application module will make sure that a Working Memory database is available before proceed with the analysis. Next the module will search for rule set number to identify the sequence of rules to process. Each Expression object captured when populating rules is used to indicate whether it is currently true or false. This is achieved by displaying captured questions (attributes of the Expression Objects). When the user agrees by selecting the yes button it will give a TRUE value which is equal to one. This value is compared with the condition value and if the value is equal then it will execute the action. The module will then search for the next RULE. Figure 6-2 shows the sub-classes of the simple_resulting_action of the KRM which have been captured in the database used by KRM rules listed in table 6-6.

Results obtained by applying KRM rules listed in table 6-6 are shown by figure 6-3 (a), (b), (c ), (d), (e) and (f). The displays are split to get a better display
output. After applying all the KRM’s rules, specifications that need to be considered in term of priority are listed as depicted by figure 6-3 (f).

<table>
<thead>
<tr>
<th>Items</th>
<th>Description</th>
<th>Class Realisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruleset 1</td>
<td>Display Initial Benefit Value</td>
<td></td>
</tr>
<tr>
<td>Rule1_1</td>
<td>If HoQ database exists then Display Benefit Value</td>
<td></td>
</tr>
<tr>
<td>Ruleset2</td>
<td>Set Initial Benefit Values Action</td>
<td></td>
</tr>
<tr>
<td>Rule2_1</td>
<td>If Customer needs benchmark value of own product is greater than competitor need score value then set the benefit value for each of the specification in this row a value of zero</td>
<td>Set_initial_Benefit_Values</td>
</tr>
<tr>
<td>Rule2_2</td>
<td>If HoQ database exists then Display Benefit Value</td>
<td></td>
</tr>
<tr>
<td>Ruleset3</td>
<td>Check benefit values spec benchmark action</td>
<td></td>
</tr>
<tr>
<td>Rule3_1</td>
<td>If own product’s specification value is greater that competitor’s specification value then set this column of specification for the NEEDS object a value of zero</td>
<td>Check_benefit_values_spec_benchmark_Action</td>
</tr>
<tr>
<td>Rule3_2</td>
<td>If HoQ database exists then Display Benefit Value</td>
<td></td>
</tr>
<tr>
<td>Ruleset4</td>
<td>Check benefit values relationship action</td>
<td></td>
</tr>
<tr>
<td>Rule4_1</td>
<td>If benefit value not equal to zero then calculate benefit value</td>
<td>Check_benefit_values_Relationship_Action</td>
</tr>
<tr>
<td>Rule4_2</td>
<td>If HoQ database exists then Display Benefit Value</td>
<td></td>
</tr>
<tr>
<td>Ruleset5</td>
<td>Determine priority</td>
<td></td>
</tr>
<tr>
<td>Rule5_1</td>
<td>If Total benefit value exists then Display Total Benefit values</td>
<td>Win_Print_Benefit_Value_TOTAL_Action</td>
</tr>
<tr>
<td>Rule5_2</td>
<td>If Total benefit value ready for sorting then Display sorted total benefit value</td>
<td>Win_Print_Benefit_Value_Sorting_Action</td>
</tr>
</tbody>
</table>

Table 6-6 Rules to Determine Design Attributes Priority
Figure 6-2 Database Browser Showing an Implementation of Sub-Classes of Simple Action Class to Determine Design Feature Priority
Figure 6-3 (a) The Output for Initial Coefficient Values After Applying KRM's First Rule
**Figure 6-3 (b) The Output of Coefficient Values After Applying KRM’s Second Rule**

Assessment of The QFD Information Model
Figure 6-3 (c) The Output for Coefficient Values After Applying KRM's Third Rule

Assessment of The QFD Information Model
Figure 6-3 (d) The Output of Coefficient Values After Applying KRM's Fourth Rule
Figure 6-3(e) The Output for Coefficient Values After Applying KRM's Fifth Rule
In conclusion, using the KRM’s rules depicted by table 6-6 to analyse the information of a mug captured by the QFD information model, the most significant specifications that need specific attention are: size and cross-sectional thickness of the HANDLE. Priority for further work or modification should be given to these two specifications before other specifications listed in figure 6-3 (f) are considered.
6.3 CASE STUDY 2: MID-RANGE HI-FI SYSTEM DESIGN

As described in chapter 3, this research assumed that customers' needs are already documented and has not focused on methods of collecting raw data for analysis. The data presented in this second case study has been published [Fung, 1997] but used in a different context. It is a mid-range hi-fi equipment, a common household appliance.

It is generally believed that this hi-fi equipment has a large user-base and its functions and features are easy to comprehend. However, to meet the software requirement, technical benchmark values of the product being analysed have been fabricated for demonstration purposes.

6.3.1 Populating the Information in the Product Model Database

The information presented has been extracted and split into customer needs and design features sections as depicted by table 6-7 and table 6-8, populated using methods described in section 4.4.3.

Interrelationship and correlation matrices are populated and assigned weighting values using methods described in section 4.4.3 and section 4.4.2 respectively. Similarly, customers' needs and technical assessment benchmarkings are populated using methods described in section 4.4.3.
### Chapter 6

#### Competitive Benchmark

<table>
<thead>
<tr>
<th>Primary need</th>
<th>Secondary need</th>
<th>Important Rating</th>
<th>Own Product</th>
<th>Competitor A</th>
<th>Competitor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good Sound</td>
<td>Strong Bass</td>
<td>6.8</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Reality Low Loss and Noise</td>
<td>16.2</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Natural Sound</td>
<td>4.7</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Clear and distinctive</td>
<td>2.0</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>More</td>
<td>Contain multi-disc drive</td>
<td>6.3</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Functional Feature</td>
<td>Contain Basic Functional Features</td>
<td>11.8</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Sufficient Memory</td>
<td>2.3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Compatible to other system</td>
<td>1.2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>More loud Speaker Level</td>
<td>2.1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>More Sound</td>
<td>More Surround Mode</td>
<td>3.9</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Features</td>
<td>Large Output Power</td>
<td>3.3</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Have Effective Mechanism to Reduce Noise</td>
<td>1.4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Performance Consistency in Playing all types of Music</td>
<td>0.5</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>More Group of Speakers can be connected to the system</td>
<td>1.6</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>More equaliser Bands</td>
<td>3.3</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*Table 6-7 The Information of Customers’ Needs and Competitive Benchmark Values for a Mid-Range Hi-Fi Equipment [Fung, 1997]*
<table>
<thead>
<tr>
<th>Primary need</th>
<th>Secondary need</th>
<th>Important Rating</th>
<th>Own Product</th>
<th>Competitor A</th>
<th>Competitor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good System</td>
<td>Reliable</td>
<td>5.8</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Performance</td>
<td>Lower Power Consumption</td>
<td>1.1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Fast Response</td>
<td>2.5</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Easy to use/</td>
<td>Full Remote Control</td>
<td>2.5</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Control</td>
<td>Clear Instruction on Keys</td>
<td>1.0</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Clear and Aesthetic Display</td>
<td>2.2</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Simple Operating Procedures</td>
<td>2.7</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Large Memory in Supporting Programming</td>
<td>0.4</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Lower Keys Input</td>
<td>0.3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Simple and Easy to Read Instruction</td>
<td>0.3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Manual</td>
<td>0.3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Aesthetic</td>
<td>Average Standard Size</td>
<td>2.9</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Fashionable Appearance and Colour</td>
<td>4.8</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>After Sale Service</td>
<td>Long Warranty Period</td>
<td>1.6</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Good Live Time Support</td>
<td>0.7</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Fast Service Response Time</td>
<td>2.0</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Short Repair Turn-round Time</td>
<td>1.6</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Low Maintenance Fee</td>
<td>2.3</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6-7 The Information of Customers' Needs and Competitive Benchmark Values for Mid-Range Hi-Fi [Fung, 1997] (cont)
## Chapter 6

### Competitive Technical Benchmark

<table>
<thead>
<tr>
<th>ASPECT</th>
<th>Specification</th>
<th>Unit</th>
<th>Measurement</th>
<th>Difficulty</th>
<th>Own Product</th>
<th>Competitor A</th>
<th>Competitor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD Player</td>
<td>No. of Disc Drive</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Loading Type</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Programming Memory</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Skipping Time (track Reading Time)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Display Mode</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Amplifier</td>
<td>Grouping of output ports</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>No. of Bands in Graphic Equaliser</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Band Width Distribution</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Power Output</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No. of Speaker Group</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>No. of choices of surround modes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>No. of pre-set modes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Tuner</td>
<td>Type of Tuning</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No. of Pre-set Channel</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Sensitivity - Easy to Tune</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Sensitivity - Amplification of noise</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6-8 The Information of Design Features and Competitive Technical Benchmark Values for a Mid-Range Hi-Fi Equipment [Fung, 1997]

119

... Assessment of QFD Information Model
### Chapter 6

#### Competitive Technical Benchmark

<table>
<thead>
<tr>
<th>ASPECT</th>
<th>Specification</th>
<th>Unit</th>
<th>Measurement</th>
<th>Difficulty</th>
<th>Own Product</th>
<th>Competitor A</th>
<th>Competitor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loud Speaker</td>
<td>No. of active units in a speaker</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Diameter of Unit</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Size of Speaker Box</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Drive Units Location in Box</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Material of Box/Drive Unit</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Roll Off (Base)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Output Power</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Cassette Deck</td>
<td>No. of Decks</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No. of Reading Heads</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Dubbing Function (speed &amp; quality)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Noise Reduction Function</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6-8 The Information of Design Features and Competitive Technical Benchmark Values for a Mid-Range Hi-Fi Equipment [Fung, 1997] (cont.)
6.3.2 The Output Display of the Hi-Fi Equipment Populated in the Product Model

The results of the captured information about Hi-Fi Equipment are shown by figure 6-4, figure 6-5 and figure 6-6 (Three figures are needed to display this volume of data). Figure 6-5 shows information about customers' needs that includes the primary needs, secondary needs and relative importance information. Notice that customer relative importance scale is different from the convention described in section 4.4.2. Figure 6-4 illustrates a portion of the design features (aspects and specifications) and correlation matrix information, whilst figures 6-6 shows information about customers' needs benchmark.

6.3.3 Analysis of the HoQ to Prioritise Design Attributes

The objective of this analysis is to demonstrate the capability of the generic KRM's rules that can be used to manipulate the captured Hi-Fi design information. Methods described in section 5.4.3 are used to analyse the information and display output are obtained in the usual way as depicted by figure 6-3 (a) - (f). Figure 6-7 shows the listing of specification in order of priority.

In conclusion, the analysis done on the Hi-Fi design information revealed that modification on specification of the design features should focus on No. of Active Units in a Speaker, Diameter of Unit and Output Power for design feature's aspect, LOUD SPEAKER before other specifications are considered.

Also, experts built to suit the mug design seem to transfer well to Hi-Fi design. This proved the capability of generic KRM's rules in supporting the analysis of the QFD information.
Figure 6-4 A Display of a Section of Specification Correlation Matrix for a Mid-Range Hi-Fi Equipment
Figure 6-5 A Display of Primary Needs, Secondary Needs and a Section of the Interrelationship Matrix for a Mid-Range Hi-Fi Equipment
### Figure 6.6 A Display of Customers' Needs Benchmark for a Mid-Range Hi-Fi Equipment

<table>
<thead>
<tr>
<th>COMPETITORS</th>
<th>COMPETITORS</th>
<th>COMPETITORS</th>
<th>SPEECH HICUST NEEDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Own Product</td>
<td>Competitor A</td>
<td>Competitor B</td>
<td>Strength/Weakness</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>Strong Bass</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>Reality Low Loss &amp; Noise</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>6</td>
<td>Natural Sound</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>Clear &amp; Distinctive</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>Content Multi-Dialog</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>Content Basic-Functional Features</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>Sufficient Memory</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>Compatible to Other System</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>More Loud Speaker Level</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>More Surround Mode</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>Large Output Power</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Effective Mechanism to Reduce Noise</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>Performance Consistency</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>More Group of Speakers</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>More Equalizer Band</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>Reliable</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>Linear Power Consumption</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>Fast Response</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>Full Remote Control</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>Clear Instruction on Keys</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>Clear &amp; Attractive Display</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Simple Operating Procedures</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>Large Memory for Programming Processor</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>Lower Keys Input</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>Simple &amp; Easy to Read Manual</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>Average Standard Size</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Fashionable Appearance &amp; Colour</td>
</tr>
</tbody>
</table>

...Assessment of QFD Information Model
<table>
<thead>
<tr>
<th>ASPECT</th>
<th>SPECIFICATION</th>
<th>VALUE COEFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUD SPEAKER</td>
<td>No. of Active Units in a Speaker</td>
<td>72.67</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Diameter of Unit</td>
<td>72.67</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Output Power</td>
<td>56.67</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>Power Output</td>
<td>45.00</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>No. of Speaker Group</td>
<td>42.00</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Roll Off (Base)</td>
<td>39.67</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>No. of Bands in Graphic Equaliser</td>
<td>37.67</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>Grouping of output ports</td>
<td>34.67</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>Band Width Distribution</td>
<td>31.33</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>No. of Disc Drives</td>
<td>22.00</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>Loading Type</td>
<td>20.00</td>
</tr>
<tr>
<td>TUNER</td>
<td>Sensitivity – Amplification of Noise</td>
<td>19.00</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>No. of Preset Modes</td>
<td>16.33</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>Programming Memory</td>
<td>10.00</td>
</tr>
<tr>
<td>TUNER</td>
<td>No. of Preset Channel</td>
<td>7.33</td>
</tr>
<tr>
<td>TUNER</td>
<td>Sensitivity – Easy to Tune</td>
<td>6.00</td>
</tr>
<tr>
<td>CASSETTE DECK</td>
<td>No. of Reading Heads</td>
<td>6.00</td>
</tr>
<tr>
<td>CASSETTE DECK</td>
<td>No. of Decks</td>
<td>4.67</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>Skipping (Track Reading) Time</td>
<td>3.00</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Drive Units Location in Box</td>
<td>0.00</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Material of Box/Drive Unit</td>
<td>0.00</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Sensitivity</td>
<td>0.00</td>
</tr>
<tr>
<td>TUNER</td>
<td>Type of Tuning</td>
<td>0.00</td>
</tr>
<tr>
<td>AMPLIFIER</td>
<td>No. of Choices of Surround Modes</td>
<td>0.00</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>Display Mode</td>
<td>0.00</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>Size of Speaker Box</td>
<td>0.00</td>
</tr>
<tr>
<td>CASSETTE DECK</td>
<td>Dubbing Function</td>
<td>0.00</td>
</tr>
<tr>
<td>CASSETTE DECK</td>
<td>Noise Reduction Function</td>
<td>0.00</td>
</tr>
</tbody>
</table>

*Figure 6-7 A Display of Specification Priority Listing for a Mid-Range Hi-Fi Equipment*
6.4 SUMMARY

QFD information modelling environment that uses OO methodology can be used to decompose the HoQ information into individual elements in a well-structured way. Also, OO methodology facilitates the integration of these individual elements into a large system. Hence the use of QFD information model enables the information to be captured and stored in Product Model as established by examples from the mug and mid-range hi-fi equipment.

Once the information is stored in Product Model, this information can be manipulated in a consistent and reasoned manner as demonstrated by the capabilities of the KRM's rules. This promises to promote the potential capability of the QFD information model in the context of integrating with an intelligent knowledge-based system. This should significantly reduce the time to analyse the QFD information. Further evaluation is necessary to explore the QFD information potential and will be the main focus of chapter 7.
7

**ENHANCEMENT OF THE QFD INFORMATION MODEL AND ITS POTENTIAL BENEFITS**

7.1 INTRODUCTION

The implementation of QFD information model, and its ability to perform systematic analysis of the captured information as established in chapter 6, offers a number of important advantageous features over other QFD information modelling approaches known to the author. However to maximise the benefits that can be realised further illustration of the information structure is necessary to demonstrate its capability of providing a practical solution to contemporary users of QFD.

This chapter will exhibit how the software is further enhanced by capitalising on its information structure flexibility. It will include illustration of the structure’s reusability and maintainability. It also considers the possibility of analysing a simplified version of QFD information by utilising first level of customer needs and design features information.
7.2 QFD INFORMATION MODEL - AN INTEGRATED OBJECT ORIENTED PRODUCT MODEL

QFD information structure designed using OO methodology has a flexible structure that is easily maintained and reused. This section will illustrate examples leading to the contribution of the structure’s functionality with respect to OO methodology. It describes the implementation of the QFD information structure described in chapter 4 resulting from modification of information structures created earlier, taking significant advantage from OO methodology.

7.2.1 Initial QFD Information Model

When this research started an initial QFD Information Model was proposed. In this information structure, it was decided to capture only customer secondary needs information represented by instances of the NEEDS class object into the Product Model as depicted by figure 7-1.

![Figure 7-1 The Initial QFD Information Model Structure](image-url)
However after discussions with industrial users the author was convinced that the incorporation of customer primary needs information into the existing model will be an added advantage especially to its structure’s flexibility. Large amounts of qualitative data can be aggregated into sub-groups based on similarities between items [Shillito, 1994; Griffin et al, 1992; Morrell, 1988]. The aggregation process creates a more information-rich model by clustering similar items and later assign a heading to the cluster, hence promoting a general level of understanding, and exploring hidden relationships [Shillito, 1994]. The improved version of the QFD information model has been described in greater detail in chapter 4.

7.2.2 The Implementation of the Modify QFD Information Structure

The following section will illustrate how the initial QFD information structure is modified, producing a more efficient and practical information structure as established in chapter 4. As the design of the structure is implemented using OO methodology, the process of implementing the new structure becomes simple. Two important steps were addressed.

(a) Modification of the Class Diagram.
(b) Modification of the Database Schemas.

7.2.2.1 Modification of the Class Diagram

The initial information structure described by figure 7-1 is maintained and used for the implementation. Since the additional information needed by the structure is only about customer primary need it can be easily added into existing structure, still maintaining its structural integrity. The process involved adding four additional classes namely, PRIMARY_NEEDS, COMPETITOR_PRIMARY_NEEDSCORE, PRIMARY_NEED_SPEC_INTER
RELATION and PRIMARY_NEEDS_ASPECT_INTERRELATION classes into the initial structure.

A relationship of PRIMARY_NEEDS class is first added to PRODUCT class and the NEEDS class detached from the PRODUCT class is attached to this new PRIMARY_NEEDS class. Following this modification, the other three classes are added and attached to their specific positions based on their relationship. The process is best described by figure 7-2.

Figure 7-2 The PRIMARY_NEEDS Class and its Associates Added to the Initial QFD Information Structure
The clouds with lighter colour represent the four additional new classes added to the initial QFD information structure. Notice that the relationship between PRODUCT and NEEDS classes has been detached to allow for the additional PRIMARY_NEEDS class. One end of the relationship of the newly added class is attached to PRODUCT class and the other end is attached to NEEDS class to complete the circuit. Three other classes associated to PRIMARY_NEEDS are also added to the system in a similar manner.

7.2.2.2 Modification of the Database Schemas

The previous section has shown one of the powerful features available in OO design. It demonstrates the ability of OO design to distinguish objects, add them or removed any unwanted object without affecting its neighbouring objects. Another important feature found in OO methodology is its principle of modularity. Modularity helps the additional program to be compiled separately and later interfaced using a header file created from the compilation. This explains why the author has decomposed the compilation of the classes into four separate data definition language files (ddl) of which the additional PRIMARY_NEEDS class is compiled in a separate file called primary_needs.ddl. Figure 7-3(a) and (b) described how the code written in C++ was modified on PRODUCT class. Notice that the existing code changes involved only on replacing the appropriate classes to the handle of the database. Similarly, changes are also reflected on method of the respective classes.

```cpp
class product_aro: public ooObj{
private:
    char ooHandle(NEEDS_aro) ooHandle(ASPECTS_aro)
public:
    product_aro(char* thename);
    ...;
    ooHandle(NEEDS_aro) get_needs(char* needs_name);
    the_product[50];
    theNEEDS[ ]<-> one_prod;
    someaspects[ ]<-> thisproduct;
}
```

*Figure 7-3 (a) Initial Implementation Part of the Coding for PRODUCT Class in ddl File*
7.2.3 Future Expansion of the Software

The existing class structure can be extended in a similar manner, to include other classes derived from the remaining phases of the QFD system. Some of the classes are already available in the existing structure, like for instance ASPECT and SPECIFICATION classes, can be used as inputs for the second phase of HoQ. Besides building the classes around these two classes, most of the schema written for the existing structure can be duplicated and reused with modification. The modifications include changing class names and possibly adding some methods. New modules can be created to house new classes in addition to the existing classes.

7.3 ENHANCEMENT OF THE CAPTURED QFD INFORMATION IN THE PRODUCT MODEL

The use of QFD information modelling during product design, supported by the OO methodology has proved beneficial. This is because it is capable of establishing a new object or objects from existing structure with little modification without affecting the coding of the main body of other classes in the structure. This is particularly useful when modelling a large system like QFD.
On the application side, the software has also proved that it is capable of analysing the captured information systematically as demonstrated by the two examples. In these examples QFD information captured as persistent objects in the Product Model is again utilised without having to make any alteration in the class structure. This is particularly useful when designers wish to have a quick analysis on the performance of their product based on the aspect or primary need. As established previously, information about the product has already been captured as persistent objects in the Product Model, hence the task focuses on manipulating the information. Rules must be captured as instances of the KRM to provide the support for this analysis.

7.3.1 Example 1: QFD Information Enhancement on Mug Design

The main objective of this exercise is to identify ASPECTs that require modification in order of priority. In this example, information about customers' needs (primary and secondary needs) and aspects as described by table 6-1 and table 6-2 respectively is used to create the HoQ. There is also other additional information required that includes interrelation of the customers' needs and aspects, correlation between aspects and aspect competitive benchmarks. Information about aspect correlations is captured in the usual way as described in chapter 4 and information about interrelation of the customers' needs and aspects, and aspect competitive benchmarks has been populated using rules available in the KRM (described in section 5.4.2). Output display of this HoQ is depicted by figure 7-4. Similarly, rules populated to determine aspect attributes priority (depicted by table 6-6) are used and the results are shown by figure 7-5 (a), figure 7-5 (b), figure 7-5 (c), figure 7-5(d), figure 7-5 (e) and figure 7-5 (f).

In this example, the result obtained indicates that if the designer wishes to make a modification on design features' aspect, the most appropriate choice in order of its priority would be HANDLE and CONTAINER as depicted by figure 7-5(f).
Figure 7-4 The Display Output of the Focused HoQ of a Mug
Chapter 7

### Figure 7-5 (a) The Output Display of Initial Coefficient Values for Aspect Priority Analysis of a Mug

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUF. CAPABILITY</th>
<th>SURFACE DECORATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe to Use</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>Durability</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Carry</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Usable Heat Transfer</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Reasonable Price</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Does not Collect Dirt on Surface</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Clean</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
</tbody>
</table>

### Figure 7-5 (b) The Output Display of Coefficient Values After Applying the Second KRM's Rule for Aspect Priority Analysis of a Mug

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUF. CAPABILITY</th>
<th>SURFACE DECORATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe to Use</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>Durability</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Carry</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Usable Heat Transfer</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Reasonable Price</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Does not Collect Dirt on Surface</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Clean</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
</tbody>
</table>

### Figure 7-5 (c) The Output Display of Coefficient Values After applying the Third KRM's Rule for Aspect Priority Analysis of a Mug

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUF. CAPABILITY</th>
<th>SURFACE DECORATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe to Use</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>Durability</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Carry</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Usable Heat Transfer</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Reasonable Price</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Does not Collect Dirt on Surface</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
<tr>
<td>Easy to Clean</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-0.20</td>
</tr>
</tbody>
</table>

Enhancement of the QFD Information Model
### Chapter 7

#### Figure 7-5 (d) The Output Display of Coefficient Values After Applying the Fourth KRM’s Rule for Aspect Priority Analysis of a Mug

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUF. CAPABILITY</th>
<th>SURFACE DECORATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to Use</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Minimal Heat Transfer</td>
<td>2.15</td>
<td>1.45</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Easy to Carry</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Easy to Hold</td>
<td>2.25</td>
<td>4.57</td>
<td>0.09</td>
<td>0.00</td>
</tr>
<tr>
<td>Visually Attractive</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Reasonable Price</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Does not Collect Dirt on Surface</td>
<td>0.07</td>
<td>1.28</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Easy to Clean</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Figure 7-5 (e) The Output Display of Coefficient Values After Applying the Fifth KRM’s Rule for Aspect Priority Analysis of a Mug

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CONTAINER</th>
<th>HANDLE</th>
<th>MANUF. CAPABILITY</th>
<th>SURFACE DECORATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Handle</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Manuf. Capability</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Surface Decoration</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Figure 7-5 (f) The Output Display of Coefficient Values of a Mug Listed in Order of Priority

<table>
<thead>
<tr>
<th>ASPECT</th>
<th>SPECIFICATION</th>
<th>VALUE COEFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td></td>
<td>7.30</td>
</tr>
<tr>
<td>Container</td>
<td></td>
<td>5.31</td>
</tr>
<tr>
<td>Manuf. Capability</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>Surface Decoration</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

...Enhancement of the QFD Information Model

136
7.3.2 Example 2: QFD Information Enhancement on Mid-Range Hi-Fi Equipment

The main objective of this exercise is to identify ASPECTs that require modification in order of priority using a consolidated HoQ information. In this example, information about primary needs and aspects as listed by table 6-1 and table 6-2 is used to create the consolidated HoQ. Besides using the existing information such as primary needs and aspects, additional information required includes interrelation of customers’ primary needs and aspects, aspect correlations, aspect competitive benchmarks, and primary need competitive benchmarks.

Aspect correlation information is captured in the usual way as described in chapter 4. As for aspect competitive benchmarks, it is populated using the KRM rule described in section 5.4.2. Similarly, interrelation matrix and primary need competitive benchmark is captured using the rule available in the KRM that adopts the same principle with the rule described in section 5.4.2.

The output display of the consolidated HoQ is depicted by figure 7-6. Notice that weightage values for interrelation matrix and competitive benchmarks shown by the display output do not follow the convention described by table 4-6 and 4-7. This is due to rules applied in the KRM that calculates the average of the respective weightage values.

Aspects of Design Features’ Priority are determined using KRM rules described in section 5.4.3. Figure 7-7 (a) and figure 7-7 (b) show results after applying the KRM’s rules. It only shows the portion of the output after applying KRM’s fifth rule. In figure 7-7 (b) the display output listed aspects in order of priority if designers wish to modify aspects of designer features. The most probable choice would be AMPLIFIER and LOUD SPEAKER.
Chapter 7

Figure 7-6 An Output Display of the Consolidated HoQ of a Mid-Range Hi-Fi Equipment

<table>
<thead>
<tr>
<th>PRIMARY NEEDS</th>
<th>CD PLAYER</th>
<th>AMPLIFIERS</th>
<th>TUNER</th>
<th>LOUD SPEAKER</th>
<th>CASSETTE DECK</th>
<th>COMPETITOR A</th>
<th>COMPETITOR B</th>
<th>PRIMARY NEEDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good Sound</td>
<td>NA</td>
<td>0.06</td>
<td>1.00</td>
<td>1.00</td>
<td>2.00</td>
<td>0.81</td>
<td>2.00</td>
<td>3.35</td>
</tr>
<tr>
<td>More Functional Features</td>
<td>NA</td>
<td>0.94</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>Good Sound Features</td>
<td>NA</td>
<td>0.06</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.81</td>
<td>2.00</td>
<td>3.35</td>
</tr>
<tr>
<td>Good System Performance</td>
<td>NA</td>
<td>0.00</td>
<td>0.20</td>
<td>0.00</td>
<td>1.00</td>
<td>0.50</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>Easy to Use/Control</td>
<td>NA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.81</td>
<td>2.00</td>
<td>3.35</td>
</tr>
<tr>
<td>Aesthetic</td>
<td>NA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.81</td>
<td>2.00</td>
<td>3.35</td>
</tr>
</tbody>
</table>

**Figure 7-6** An Output Display of the Consolidated HoQ of a Mid-Range Hi-Fi Equipment
### Figure 7-7 (a) An Output Display of the Consolidated HoQ Coefficient Values for a Mid-Range Hi-Fi Equipment

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>CD PLAYER</th>
<th>AMPLIFIER</th>
<th>TUNER</th>
<th>LOUD SPEAKER</th>
<th>CASSETTE DECK</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY NEED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good Sound</td>
<td>0.00</td>
<td>1.96</td>
<td>1.00</td>
<td>2.63</td>
<td>0.00</td>
</tr>
<tr>
<td>More Functional Features</td>
<td>1.64</td>
<td>1.14</td>
<td>0.86</td>
<td>1.05</td>
<td>0.00</td>
</tr>
<tr>
<td>More Sound Features</td>
<td>0.00</td>
<td>1.64</td>
<td>0.38</td>
<td>0.42</td>
<td>0.00</td>
</tr>
<tr>
<td>Good System Performance</td>
<td>0.60</td>
<td>0.10</td>
<td>0.26</td>
<td>0.13</td>
<td>0.00</td>
</tr>
<tr>
<td>Easy to Use/Control</td>
<td>0.60</td>
<td>0.20</td>
<td>0.39</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Aesthetic</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>TOTAL BENEFIT VALUES</td>
<td>2.24</td>
<td>6.95</td>
<td>2.87</td>
<td>4.22</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Figure 7-7 (b) An Output Display of the Consolidated Coefficient Values for a Mid-Range Hi-Fi Equipment Listed in Order of Priority

<table>
<thead>
<tr>
<th>ASPECT</th>
<th>VALUE COEFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMPLIFIER</td>
<td>6.05</td>
</tr>
<tr>
<td>LOUD SPEAKER</td>
<td>4.22</td>
</tr>
<tr>
<td>TUNER</td>
<td>2.67</td>
</tr>
<tr>
<td>CD PLAYER</td>
<td>2.24</td>
</tr>
<tr>
<td>CASSETTE DECK</td>
<td>0.00</td>
</tr>
</tbody>
</table>

---

139 Enhancement of the QFD Information Model
7.4 BENEFITS FROM USING THE QFD MODEL AND ITS ENHANCED APPLICATION

The QFD information model structure aims to support the design process in a consistent way. During the design phase, a complete description of the QFD activities can be modelled using the QFD information model. Also, the QFD information model offers the persistent objects (residing within the Product Model) with an associated graphical user interface and knowledge base analysis capability, including the KRM. As a result the time required for developing a product can be greatly reduced. Case study examples described in chapter 6 and 7, indicated that the QFD information model is a suitable design support tool for developing a product during its design stage. The same methodology can be applied to develop the remaining stages of the QFD processes and will be the author’s main focus for future research as described in chapter 8.

The QFD information modelling capability offers inherent characteristics of an OO approach and supports modularity of message-passing connections between distinct elements of the QFD system. In this way rules for analysing the information can be incorporated, after QFD information has been captured.

In addition, the QFD information modelling methods, through the development of OO technology for developing QFD systems, leads to the following benefits:

7.4.1 Flexibility Enhancement

By decomposing elements of the HoQ into an appropriate set of object classes, improved flexibility facilitates the realisation of specific application needs. This is enabled by facilitating the addition, deletion or modification of new modules during the modelling process. Data for each new element is also easily added, deleted or modified in the relevant data model without the need to change other existing data. Furthermore, no changes are required to the source code that drive the simulation.
7.4.2 Efficient Development of Software and its Application

The QFD information models of each element of the HoQ can be connected together, along with the KRM to produce a complete QFD information model system, that has the capability to represent the QFD system in a consistent and formal way. Thus, QFD information modelling approach offers means of handling concurrency, modularity, reconfigurability and reusability.

One advantage to be gained (in the examples described in previous sections) from this model, is the ability to experiment alternative combination between primary customer needs or secondary customer, and aspects or specifications. The approach can reduced lead times associated with product design development, by enabling integration software to be independently developed and tested.

7.5 SUMMARY

Thus the Product Data Model approach conceived by the author demonstrates additional advantages over and above those commonly realised. Deployment and development of the QFD information modelling and simulation environment has demonstrated the enhanced capability to provide a designer with a support for development of a product during the design stage. In comparison with other approaches (Bird, 1992; Sriraman, 1990), it has been demonstrated that the modelling approach developed is easier to understand, produces better structured models, and can reduce modelling complexity, especially when developing and analysing large systems. The flexibility of the QFD information model approach has also been demonstrated in terms of its responsiveness to change, that it can be easily adapted to support the addition, deletion and modification of the object classes.
Further enhancement was the use of the alternative combination of the HoQ elements supported by the rules in the KRM, eased its implementation. Hence, the integrated used of QFD information model and KRM has the capability to develop the state of art QFD system, including means of effectively co-ordinating the implementation of the remaining phases of the QFD system.
CONTRIBUTIONS TO KNOWLEDGE AND ISSUES FOR FURTHER INVESTIGATION

8

8.1 INTRODUCTION

This chapter summarises the author's contributions to knowledge through the development and deployment of information structures and modelling approaches. Potentially, methods of the type proposed and investigated could impact on the way QFD information is captured and systematically managed despite the growing complexity of QFD applications. The author's research has also identified activities in which future research is required before the model architecture can be commercially available.

8.2 CONTRIBUTIONS TO KNOWLEDGE

8.2.1 QFD Information Founded upon a Product Model

A central focus of the thesis has been the development of techniques which help to realise a comprehensive implementation capturing QFD information as established in
Chapter 4. QFD information has been modelled as an information structure. This has led to an improved understanding of factors involved in capturing QFD information. The research has focused on structuring the first phase of the QFD system (the HoQ) as it is the main thrust of QFD upon which later phases are built. OO methodology has been chosen to model the behavioural characteristics of the QFD processes. The author has identified and developed suitable mechanisms (as described in chapter 4) that enable QFD information to be captured together with supporting computing facilities. In this context, the information is captured as persistent objects resident in a database as part of a Product Data Model.

In addition, the author has carried out two case studies (described in chapter 6) which demonstrate the use of the information model. This includes the capture of information about a mug, which led to a knowledge of processing a simple consumer product as a pilot software prototype assessment; the extended demonstration from the second case study demonstrates the ability to facilitate an industrial based application.

The information modelling so derived could impact current practice in two ways,

1. By providing a means of systematic capture of information about QFD activities with interactive user interface supports.

2. By creating persistent objects stored in Product Model which can be readily accessed when required.

Hence information about customers' needs can now be made available in the Product Model and can be exploited at the very first crucial stage of the design process. Such information which supports the design process, is considered to be an important requirement that provides the designer with a better understanding about effects of customer desire in achieving a competitive and value added product.
8.2.2 Knowledge-based QFD Expert Analysis

The author's research has also established a knowledge-based system capable of providing support that enables the QFD information captured within the Product Model to be analysed. An intelligent knowledge-based system has been developed by integration with the structure of the Knowledge Representation Model (KRM) developed from previous research [Harding, 1996b], to verify the flexibility of manipulating the captured information available in the Product Model. In this context, the study generated rules that are captured as instances of the KRM (described in chapter 5) to support the analysis. The rules captured are generic in the way they can be exploited to analyse other product attributes captured by the QFD information.

Potentially the intelligent knowledge-based QFD design experts could impact the way in which complex QFD information is analysed. Here the established KRM, which underpins the capturing of rules, has been used to accomplish useful, consistent, reasoned analysis of the QFD information. In this way, the current QFD processes which are completely manual or aided via a static documentation by a computer system can benefit from the research findings. This is indeed a step closer to realising a comprehensive QFD computer system that can automatically check consistency and enable associativity with CAD systems and other engineering databases.

The author acknowledges that the mathematical algorithm used as rules that are populated in the KRM database do not exhibit the ideal model in determining the design feature priority (be it aspect or specification). Nevertheless it is enough to demonstrate its capability of manipulating and analysing the captured QFD information in whichever ways the designers wish, to get the required results. Hence, the author believes that other algorithms can be developed from AI methods such as neural network and fuzzy logic sets populated as captured knowledge in the KRM can be used to analyse the QFD information. Such AI methods that support the decision process, during the process of analysing QFD information will benefit the design team in the way that this systematic and rigorous approach of capturing knowledge makes possible.
8.2.3 Architecture of QFD Modelling System

The author has created the QFD information model based on OO methodology which has properties of being able to decompose QFD elements into abstractions of object classes (described in chapter 4). The compilation of these object classes has created separate modules that partition the application into functional systems and subsystems. Each of these has the capability of executing its function individually. This modelling approach possesses desirable characteristics such as flexibility, modularity, reusability and maintainability.

This is considered to be an important achievement as modifications to the information structure can be made on individual files without affecting the whole information structure (described in chapter 7). Using this approach, considerable development time can be saved in respect to modification of the structure and recompiling of the codes each time any alteration is made.

In addition to the above, the author has carried out experiments which demonstrate the manipulation of the information in the database. The enhanced capabilities (described in chapter 7) show the flexibility of analysing information, once it is captured in the database as persistent objects. Such enhancement can enable designers to realise appropriate performance of the QFD information model and explore its potential capabilities.

8.2.4 Integration of the QFD Information Model With Other Information Models

The author's work involved the development of QFD information model. The QFD information model resides within the Product Model. In this way, the captured information (persistent objects) can be made available for use by other design agents or design experts. This includes monitoring of the design process against the captured QFD information. The information model structure could impact on current practice by
benefiting the availability of the information integrated in the database. This integrated information system available during the design process should be able to provide a variety of information to support the design team during product development.

8.2.5 Promotion of Product Innovation

A lot of criticism of the QFD method is focused on the argument that QFD prevents innovation, or does not support innovation, in some sense preserving the technology level of the company [Gustafsson et al, 1994]. The QFD information model can be looked upon as a supporting tool to help the development team to co-operate and reach consensus decisions. The HoQ is not meant to serve as problem solving tool, but it supports the structuring of large amounts of information and in that way helps the team see where efforts should be made.

QFD decision support systems can be useful tools to help the decision-maker in the strategic decision-making process. The large amount of structured information gathered in the HoQ can be used to make decisions about the new product. The matrix can show weak points where new innovations are needed to meet the customer expectations as well as strong points where the existing solution only needs to be refined. The information that comes out of the matrix can indicate if there is a need for a totally new product or a redesign of the old product (see chapter 6 and chapter 7).

The development of the QFD information model indirectly contributes in the way that the information can made available when it is required through the mechanism as exemplified case studies described in chapter 6 and chapter 7.
8.3 RECOMMENDATIONS FOR FUTURE WORK

8.3.1 Establishment of Links With Other Concurrent Engineering Systems

The computer-based QFD information model could be linked to other Concurrent Engineering Systems. In this research, the information model developed functions as an individual system and raw data is captured and stored directly into the system. Hence the integration link between the information model and other design agents was considered to be outside the scope of the author’s research. Thus, further development of the thesis findings and further experimental work is necessary to show that it is possible to integrate other design agents that will reflect the capabilities of the information system.

When information about a product is needed for development, a common starting point for the design process is to identify product definition. If this definition can be established effectively in the Product Model, the product definition can be used to described the product. Here it can be used to define the product, and the PRODUCT class (see chapter 4) of the QFD information model will inherit this definition. Moreover, since the author’s approach to integrating systems is driven by messages throughout the process of capturing information, it may be appropriate to focus transferring of the information to a computer-aided design system to produce a graphical form of the product.

Other interface design issues concern the monitoring of the captured information within the environment: how to detect changes or conflicts occurring in the product design, and identify and inform interested design agents about the detected changes. The integration of the QFD information model with Engineering Moderator developed for this purpose [Harding, 1996a] for instance, will ensure such monitoring. Thus, the QFD information model structure provides a basis from which further research could lead to much improved product design supporting tools.
8.3.2 Enhancement of the Intelligent Knowledge-based System

The knowledge-based system developed during this research based on the KRM [Harding, 1996b] (see chapter 5), enables expert knowledge to be captured as instances of the KRM using a production rule approach. Here the author, by developing rules based on simple mathematical algorithm to determine the design feature priority, demonstrated the flexibility of the information model being manipulated and analysed. However, as the information structure is implemented using OO design, it allows the captured knowledge to be stored as RULE objects within databases. This is considered by the author to be an important line of future research to meet the emerging need for more flexible integration of other knowledge paradigms, including for example, neural networks or genetic algorithm.

Hence, potentially the integration of a more complex knowledge-based algorithms can have increased use of the database. Knowledge can be modelled using other knowledge paradigms and be integrated into the system as instances of the KRM. The author’s information model structure coupled with the KRM structure supports the process of this knowledge as the signal is passed and updates any modification made to the database. This information structure offers a conducive simulation environment that can be utilised and further developed to enhance its capability.

8.3.3 Enhancement of the Generic Intelligent Knowledge-based System

The KRM developed during this research was embodied into the information architecture, which impacts on the way in which the QFD information is manipulated and analysed. Rules populated as attributes of KRM are utilised to determine design features’ priority. These generic KRM’s rules enable analysis of different products using the same rules, as demonstrated by the examples of two case studies described in chapter 6. In these examples, experts built to suit mug design transfer well to Hi-Fi Equipment design.
Similarly, this approach can be extended when unifying other knowledge-based systems into the information architecture framework. This can simplify the process involved in manipulating and analysing the information as time can be saved on duplicating the process of populating KRM's rules. Thus by developing the generic knowledge-based system it should allow the designer to effectively use the rules without having to design specific rules for analysis of different products.

8.3.4 Enhancement of the Graphical User Interface Capability

The implementation to capture information uses primitive graphical user interface developed in a Motif environment. Since the development of this interface is outside the scope of this research, little effort has been allocated to design an optimum interface. The author has found some difficulties in displaying the traditional shape of the HoQ. However, it might be better if the system is integrated to a commercially available graphical user interface. Moreover, the time needed to design the interactive HoQ is likely to be significantly less than the time needed to write the programs. Hence the use of this interactive package would allow the interactive design of the HoQ to be more effective and aesthetically it will be very much more appreciated.

8.3.5 The Need for Improved Database System

The database (Objectivity/DB) is built as a data model to store captured information. The facilities available in the database management system are limited to capturing and storing of information. The database used is an early version of the new object oriented technology. Its application is unstable and it is now no longer supported by its vendors. However to effectively manage the database a more suitable database system should be exploited. Among important feature that must be available includes the data structure query language (SQL) so that the OO database can independently creates, queries, modifies and deletes information without having the trouble of coding.
it in the database schemas. Hence the implementation of this new database developed as an integrated system of the information model could provide a much stable database management system.

8.3.6 Development Information Structure for the Remaining QFD Phases

The research has developed a comprehensive structure that modelled the HoQ to capture information about QFD activities in the first phase (described in chapter 2). The principles underlying the HoQ apply to any effort to establish clear relations between manufacturing functions and customer satisfaction [Hauser et al., 1988]. Hence the developed information structure can be utilised to develop the next HoQ phase, that is mainly concerned with detailed product design. The class structure of design features (that become the rows) will underpin the development of this second HoQ. This process continues to a third and fourth phase as the columns of one stage become the rows of the next.

The development of the model that links the houses will implicitly convey the voice of the customer through manufacturing. Thus, developing the four phases of the HoQ with the ability to analyse the captured information will ultimately be an important asset to an enterprise. This is particularly true for an enterprise to realise elegant and innovative ideas from customers to translate them into processes and eventually produce products that ensure meeting of the customers' needs. Here useful, consistent and reasoned analysis of the information is supported by built-in knowledge-based systems along each step of the HoQs implementation. Moreover a more ambitious approach is to integrate this structure with Manufacturing Model Architecture. This would allow a designer to effectively use the system from early stage of product development to the stage that the product can be manufactured.
9

SUMMARY AND CONCLUSION

9.1 SUMMARY

This study begins with the underlying premise, that customers' inputs may provide valuable insights into the implementation and effect of product design; and with the general aim of incorporating these customers' needs into CE activities. Tools that can provide support for these customers' needs to be integrated in the design of the product are essential to maximise the business benefits achievable from closer and more integrated methods of working throughout the design process.

The research traces the history of CE and investigates the methods and means of exploiting the potential capability of this CE concept. A comprehensive review of the literature on the development of CE and its ability to utilise computer technologies was reported in chapter 2. The review showed the complexity of the subject, presented the current trend in utilising software design capabilities and also reported the recent appreciation of industrialists and academicians of the real potential of CE in improving the performance of the manufacturing enterprise. This includes the opportunity for QFD to be incorporated into current CE design practice.

The introduction of QFD to promote the integration of customers' needs into product design has helped accelerate the acceptance of the 'design community' of the need to explore the implementation of CE beyond current practice. However amidst all of this progress there remains a disturbing undercurrent which has been highlighted in section 1.1 and section 2.5.3. Applying QFD is not just a matter of gathering
information of customers' needs and design features to determine design feature specifications' target values. It forms part of a total mechanism of gathering information and must have the capability to support analysis of the information especially when the magnitude of the HoQ matrix increases.

The research has attempted to deal with the complex issue of integrating customers' needs into a CE environment by formalising a QFD information model. The information model is capable of capturing attributes of the HoQs' elements and resides as persistent objects in a Product Data Model. In a conventional HoQ process, analysis of the information relies upon users being able to interpret the information manually or with the aid of a static documentation process by computer systems. Here it is suggested that users can exploit the capability of the information structure by utilising a Knowledge Representation Model, embodied into the information architecture framework. This enables the information to be manipulated and analysed.

Using the QFD information model a repository of the HoQ information is available for analysis by the Knowledge Representation Model. This has proved to be valuable, as exemplified by case studies described in chapter 6 and chapter 7. Examples from these case studies demonstrated the potential of the QFD information model to support product design. The information model is enriched by its characteristics of modularity, integratability and reusability described in chapter 7.

The author believes that the research can impact current practise in six main areas as described in chapter 8. The findings provide designers with opportunity to marshal their resources by focusing on customers' needs to achieve an enterprises' strategic goals. Finally the research recommended six areas for future work which includes enhancement of the QFD information model and expansion of the model to include the implementation of remaining phases of the QFD system. This would offers designers the opportunity to effectively use the system from early stage of product development to the stage that the product can be manufactured.
9.2 CONCLUSION

The following conclusions can be drawn from the research.

(1) The author has developed novel QFD information structures which provides an environment for QFD information content to be captured and stored as persistent objects in a Product Model. This enables the QFD information to be accessed and manipulated when required.

(2) The author has defined an intelligent knowledge-based QFD Software Expert. The development of the intelligent knowledge-based QFD Software Expert utilises the Knowledge Representation Model (KRM) architecture. This intelligent knowledge-based QFD Software Expert demonstrates the feasibility of its implementation alongside the QFD information model to provide an intelligent QFD modelling environment.

(3) This environment incorporates an architectural framework that utilises OO programming and database management, to structure the design and implementation of the QFD information. The implementation has proved that:

QFD information content can be modelled and reside in a Product Model utilising the data structures defined in the research.

This established QFD information model, founded upon a Product Model, provides an environment that has the potential to enable all design agents to share QFD information throughout the design process.
The author has also structured the QFD information model to be accessible to a knowledge-based model. This implementation has proved that:

Intelligent knowledge-based QFD experts can accomplish useful, consistent, reasoned analysis of QFD information contained in a Product Model.

The author's modelling methods offer a starting point for further research into the provision of comprehensive support for the implementation of QFD to support CE systems. It offers methods and structures which could be used to harness the potential capabilities of QFD as a tool to support the product design process. The future application of this modelling technique may lead to important technical and commercial benefits. Thus the QFD information model has potential for commercialisation and industrial applications.
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Bibliography

Bibliography on Quality Function Deployment


Ackerman, M. & Buckland, B., 1994, 'Multiple Matrices for a Marketing QFD', 6th Symposium on Quality Function Deployment, Novi, Michigan, pp. 25 - 34.


Bibliography

*industrialised Housing application*, IIE Transactions, Vol. 26, No. 4, pp. 72-79.


Bibliography


Bibliography


Bibliography

Biondo, B, 1991, 'Application of QFD and Other Quality Tools to a Truck System', 3rd Symposium on Quality Function Deployment, Novi, Michigan, pp. 503


Brass, R.L., 1994, 'Quality Elements to Consider in Deriving the Voice of the
Bibliography


Bibliography


Bibliography


Clifton, L. H., 1993, 'Improve the Quality of Quality', Quality Progress, Vol. 26, pp. 41 - 44.


Bibliography


Bibliography


Dika, R.J., 1993, 'QFD Implementation at Chrysler - The First Seven Years', 5th Symposium on Quality Function Deployment, Novi, Michigan, pp. 307 - 326.


Bibliography


Editorial, 1994, 'Proactive Quality Control is the Key to Profitability in the 1990s', Modern Casting, Vol. 84, pp. 44.


178
Bibliography


Eureka, W. E., 1987, 'Introduction to Quality Function Deployment', ASQC Automobile Section’s Annual Quality and Reliability Conference.
Bibliography


Bibliography


Gibson, J., 1995, ‘Happy Feet, Part II: The Return of the Princeton Foot Clinic (or the QFD “Viral Strategy”)’, 7th Symposium on Quality Function Deployment,
Bibliography

Novi, Michigan, pp. 123 - 140.


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Lyman, D., 1995, 'The Balancing of QFD Matrices The Key to Understanding Your Customers' Needs', 7th Symposium on Quality Function Deployment, Novi, Michigan, pp. 75 - 84.
Bibliography

Lyman, D. & Richter, K., 1993, 'QFD and Personality Type The Key to Team Energy and Effectiveness', 5th Symposium on Quality Function Deployment, Novi, Michigan, pp. 197 - 205.


Bibliography


Bibliography


Bibliography


Miller, C., 1994, 'Using QFD to Improve Process of Automobile Painting', 6th
Bibliography


Bibliography


Bibliography


Oswald, T.H., 1993, 'QFD : A Step-change Planning Tool for Engineering and Construction Projects', 5th Symposium on Quality Function Deployment, Novi,
Bibliography

Michigan, pp. 237 - 244.


Bibliography


Bibliography


Robertshaw, W. G., 1995, 'Using an Objective Sales Point Measure to Incorporate Elements of the Kano Model into QFD', 7th Symposium on Quality Function Deployment, Novi, Michigan, pp. 201 - 216.


Ross, H. & Paryani, K., 1995, 'QFD Status in the W.S. Automobile Industry', 7th
Bibliography

Symposium on Quality Function Deployment, Novi, Michigan, pp. 575 - 584.

Ross, P.J., 1988, 'Role of Taguchi Methods and Design of Experiments in QFD', Quality Progress, Vol. 21, No. 6, pp. 41 - 47.


Bibliography


Bibliography


Bibliography


Bibliography


Taylor, K. G., 1994, 'How to Make a Success of Quality Function Deployment', IEE Colloquium (Digest), No. 086, pp. 4/1 - 4/5.


Bibliography


Ungvari, S. F., 1992, ‘To QFD or Not QFD (That is the Question)’, 4th Symposium on Quality Function Deployment, Novi, Michigan, pp. 67 - 103.


Bibliography


Bibliography


Wu, B. C., 1992, 'Understanding Total Quality Management (TQM) and Accept the Challenge', SAVE Proceedings (Society of American Value Engineers), Vol. 27, pp. 91 - 100.


Bibliography

213 - 226.


Bibliography


Bibliography on Concurrent Engineering, Programming Language, Engineering Moderator and Manufacturing Modelling.


Bibliography


Appendix I

// The code is written by A.R.Omar
// Rev 1.0 dated 17-5-96
// The needs_spec.ddl file
// The objective is to convert customer needs to
// specification

class product_aro: public ooObj
{
  private:
    char the_product[50];

  public:
    ooHandle(primary_needs) thePrimaryNEEDS[] <-> a_prod;
    ooHandle(ASPECTS_aro) someaspects[] <-> thisproduct;
    product_aro(char* thename); // for construction by hand
    char* get_product(void);
    void put_product(char* product);
    void readin(void);
    ooHandle(primary_needs) get_needs(char* needs_name);
    void add_another_needs(ooHandle(primary_needs)
      associated_needs);
    ooHandle(ASPECTS_aro) get_aspects(char* aspects_name);
    void add_another_aspects(ooHandle(ASPECTS_aro)
      associated_aspects);
};

// The needs class primary function is to store secondary
// customer needs that a primary customer need have.

class NEEDS_aro: ooObj
{
  private:
    char the_need[200];
    int NEEDS_importance;
    int NEEDSPEC_relativeImportance;
    int NEEDASPECT_relativeImportance;
    int NEEDS_Scorecompetitor;

  public:
    ooHandle(primary_needs) a_one_need <->
      thePrimary_needs[];
    ooHandle(SPEC_INTERRELATION) theSPEC_INTER[] <->
      theneed;
    ooHandle(ASPECTS_INTERRELATION) theASPECT_INTER[] <->
      thisneed;
    ooHandle(COMPETITOR_needScore) theCOM_SPECSCORE[] <->
      oneneed;
    ooHandle(VALUE_COEFF) needvalue_coeff[] <-> a_need;

    ...needs_spec.ddl
Appendix I

NEEDS_aro();
NEEDS_aro(char* needname, int the_row);
int getNEEDS_importance(void);
void putNEEDS_importance(int importance_rating);
int getNEEDSPEC_relativeImportance(void);
void putNEEDSPEC_relativeImportance(int
specrelativeImportance_sum);
int getNEEDASPECT_relativeImportance(void);
void putNEEDASPECT_relativeImportance(int
aspectrelativeImportance_sum);
int getNEEDS_Scorecompetitor(void);
int check_need_assoc(ooHandle(COMPETITOR_needScore)
associated_need);
void putNEEDSPEC_relativelmportance(int
specrelativelmportance_sum);
void putNEEDASPECT_relativelmportance(int
aspectrelativelmportance_sum);
int getNEEDS_Scorecompetitor(void);
int check_Vcoeff_assoc(ooHandle(VALUE_COEFF)
assoc_Vcoeff);
void putNEEDS_Scorecompetitor(int scoreCompetitor);
char* get_the_needs(void);
void put_the_needs(char* needs);
void readin(void);
ooHandle(primary_needs) get_product(char* need_name);
void add_another_pr_need(ooHandle(primary_needs)
associated_need);
void put_the_needs(char* needs);
void readin(void);
ooHandle(COMPETITOR_needScore)
get_competitor_specValue(char* need_name);
void add_another_competitor_specValue
(ooHandle(COMPETITOR_needScore)
assoc_competitor_specValue);
ooHandle(SPEC_INTERRELATION) get_specInter(char*
specInter);
void add_another_specInter(ooHandle(SPEC_INTERRELATION)
assoc_specInter);
ooHandle(ASPECTS_INTERRELATION) get_aspectInter(char*
aspectInter);
void add_another_aspectInter
(ooHandle(ASPECTS_INTERRELATION) assoc_aspectInter);
int check_interaspect_assoc
(ooHandle(ASPECTS_INTERRELATION) assoc_interaspect);

II

The ASPECT class primary function is store aspect of the
II
customers' needs as defined by the design teams.

class ASPECTS_aro: ooObj
{
    private:
        char the_aspect[200];
        float aspect_importance;

...needs_spec.ddl
Appendix I

```c
public:
    ooHandle(product_aro) thisproduct <->
        someaspects[];
    ooHandle(primary_needs_aspect_interRelation)
        anASPECT_INTER[] <-> aspect_one;
    ooHandle(ASPECTS_INTERRELATION) theSPEC_INTER[] <->
        thisaspect;
    ooHandle(ASPECTS_CORRELATION) someaspect_corr1[] <->
        oneaspect1;
    ooHandle(ASPECTS_CORRELATION) someaspect_corr2[] <->
        oneaspect2;
    ooHandle(ASPECTS_CORRELATION) someaspect_corr3[] <->
        oneaspect3;
    ooHandle(COMPETITOR_ASPECTVALUE) someaspect_value[]
        <-> thisaspect;
    ooHandle(VALUE_COEFF) aspectvalue_coeff[] <->
        an_aspect;
    ooHandle(SPECIFICATION_aro) somespec[] <->
        theaspect;

ASPECTS_aro();
ASPECTS_aro(char* thename);
void put_the_aspects(char* aspect);
char* get_the_aspects(void);
float get_aspect_importance(void);
void put_aspect_importance(float aspect_importance);
void readin(void);
    ooHandle(product_aro) get_product(char* product_name);
    void add_another_product(ooHandle(product_aro)
        associated_product);
    ooHandle(SPECIFICATION_aro) get_spec(char* spec_name);
    void add_another_spec(ooHandle(SPECIFICATION_aro)
        associated_spec);
    int check_spec_assoc(ooHandle(SPECIFICATION_aro)
        associated_spec);
    int check_Vcoeff_assoc(ooHandle(VALUE_COEFF)
        assoc_Vcoeff);
    ooHandle(ASPECTS_INTERRELATION) get_aspectInter(char* 
        aspectInter_name);
    void add_another_aspectInter
        (ooHandle(ASPECTS_INTERRELATION)
            associated_aspectInter);
    ooHandle(ASPECTS_CORRELATION) get_aspectCorr2(char* 
        aspectCorr_name2);
    void add_another_aspectCorr1
        (ooHandle(ASPECTS_CORRELATION)
            associated_aspectCorr1);
    ooHandle(ASPECTS_CORRELATION) get_aspectCorr1(char* 
        aspectCorr_name1);
    void add_another_aspectCorr2
        (ooHandle(ASPECTS_CORRELATION)
            associated_aspectCorr2);
```

...needs_spec.ddl
Appendix I

```c
ooHandle(ASPECTS_CORRELATION) get_aspectCorr3(char* aspectCorr_name3);
void add_another_aspectCorr3
 (ooHandle(ASPECTS_CORRELATION) associated_aspectCorr3);
ooHandle(COMPETITOR_ASPECTVALUE)
 get_competitor_aspectValue(char* competitor_aspectValue_name);
void add_another_competitor_aspectValue
 (ooHandle(COMPETITOR_ASPECTVALUE)
 assoc_competitor_aspectValue);

// The SPECIFICATION class primary function is store
// SPECIFICATION of the customers' needs as defined by the
// design teams.

class SPECIFICATION_aro: ooObj
{
  private:
    char the_context[200];
    char the_group[200];
    char the_subgroup[200];
    char value_quality[200];
    float value_quantity;
    char unit_measurement[5];
    int spec_importance;
    int spec_target;

  public:
    ooHandle(ASPECTS_aro) theaspect <-> somespec[];
    ooHandle(primary_needs_spec_interRelation)
      aSPEC_INTER[] <-> spec_one;
    ooHandle(VALUE_COEFF) specvalue_coeff[] <-> a_spec;
    ooHandle(COMPETITOR_SPECVALUE) somespec_value[] <-
      >thisspec;
    ooHandle(SPEC_INTERRELATION) somespec_inter[] <-
      >thisspec;
    ooHandle(SPEC_CORRELATION) somespec_corr1[] <->
      onespec1;
    ooHandle(SPEC_CORRELATION) somespec_corr2[] <->
      onespec2;
    SPECIFICATION_aro();
    SPECIFICATION_aro(char* specname);
    char* get_the_group(void);
    void put_the_group(char* group);
    char* get_the_subgroup(void);
    void put_the_subgroup(char* subgroup);

216
...needs_spec.dll
```
Appendix I

char* get_the_context(void);
void put_the_context(char* context);
char* get_value_quality(void);
void put_value_quality(char* quality);
float get_value_quantity(void);
void put_value_quantity(float quantity);
char* get_unit_measurement(void);
void put_unit_measurement(char* unit);
int get_spec_importance(void);
void put_spec_importance(int spec_importance);
int get_spec_target(void);
void put_spec_target(int spec_target);
void readin(void);

ooHandle(COMPETITOR_SPECVALUE) get_specValue();

ooHandle(ASPECTS_aro) get_aspect(char* aspect_name);
void add_another_aspect(ooHandle(ASPECTS_aro)
associate_aspect);

ooHandle(SPEC_INTERRELATION) get_specInter(char*
specInter_name);
void add_another_specInter(ooHandle(SPEC_INTERRELATION)
associated_specInter);

ooHandle(SPEC_CORRELATION) get_specCorr1(char*
specCorr_name1);
void add_another_specCorr1(ooHandle(SPEC_CORRELATION)
associated_specCorr1);

ooHandle(SPEC_CORRELATION) get_specCorr2(char*
specCorr_name2);
void add_another_specCorr2(ooHandle(SPEC_CORRELATION)
associated_specCorr2);

ooHandle(COMPETITOR_SPECVALUE)
get_competitor_specValue(char*
competitor_specValue_name);
void add_another_competitor_specValue
(ooHandle(COMPETITOR_SPECVALUE)
assoc_competitor_specValue);

});
Appendix I

// The code is written by A.R.Omar
// Loughborough University

// Last updates 27-5-96
// The inter_correlate.ddl file
// The objective is a. to create interrelationship between 
// customer needs and specification 
// b. to create correlation matrix 
// between 2 specification

// The SPEC_INTER class primary function is to store the 
// relationships between customers' needs and 
// specifications that a product have.

class SPEC_INTERRELATION: ooObj{
private:
  char spec_inter[20];
  int spec_weightageValue;
public:
  ooHandle(SPECIFICATION_aro) thisspec <-> somespec_inter[];
  ooHandle(NEEDS_aro) theneed <-> theSPEC_INTER[];
  SPEC_INTERRELATION(char* intername);
  char* get_spec_inter(void);
  void put_spec_inter(char* value);
  int get_weightageValue(void);
  void put_weightageValue(int the_value);
  void readin(void);
  ooHandle(NEEDS_aro) get_need(char* need_name);
  void add_another_need(ooHandle(NEEDS_aro) associated_need);
  ooHandle(SPECIFICATION_aro) get_spec();
  void add_another_spec(ooHandle(SPECIFICATION_aro) associated_spec);
};

// The ASPECT_INTERRELATION class primary function is store 
// the relationship value between customers' need and 
// ASPECTs that a product have.

class ASPECTS_INTERRELATION: ooObj{
private:
  char the_aspect_inter[20];
  float aspect_weightageValue;
public:

Appendix I

```c
 ooHandle(NEEDS_aro) thisneed <->
     theASPECT_INTER[];
 ooHandle(ASPECTS_aro) thisaspect <-> theSPEC_INTER[];
 ASPECTS_INTERRELATION(char* thename);
 void put_aspects_inter(char* aspect_inter);
 char* get_aspects_inter(void);
 float get_weightageValue(void);
 void put_weightageValue(float the_value);
 void readin(void);
 ooHandle(NEEDS_aro) get_need(char* need_name);
 void add_another_need(ooHandle(NEEDS_aro)
     associated_need);
 ooHandle(ASPECTS_aro) get_aspect(char* aspect_name);
 ooHandle(ASPECTS_aro) get_aspect_object();
 void add_another_aspect(ooHandle(ASPECTS_aro)
     associated_aspect);

 // The SPEC_CORRELATION class primary function is store
 // the relationship value between Specifications that a
 // product have.

class SPEC_CORRELATION: ooObj
{
private:
    char the_spec_corr[20];
    int spec_weightage_corrValue;

public:
    ooHandle(SPECIFICATION_aro) onespec1 <->
        somewspec_corr1[];
    ooHandle(SPECIFICATION_aro) onespec2 <->
        somewspec_corr2[];
    SPEC_CORRELATION(char* spec_corrname);
    void put_spec_corr(char* spec_corr);
    char* get_spec_corr(void);
    int get_speccorrValue(void);
    void put_speccorrValue(int the_value);
    void readin(void);
    void readinnone(void);
    ooHandle(SPECIFICATION_aro) get_spec1(char*
        spec_name1);
    ooHandle(SPECIFICATION_aro) get_corrSpec1();
    ooHandle(SPECIFICATION_aro) get_corrSpec2();
    void add_another_spec1(ooHandle(SPECIFICATION_aro)
        associated_spec1);
    ooHandle(SPECIFICATION_aro) get_spec2(char*
        spec_name2);
    void add_another_spec2(ooHandle(SPECIFICATION_aro)
        associated_spec2);

};
```

219

...inter_correlation.ddl
Appendix I

The ASPECTS_CORRELATION class primary function is store the relationship value between Aspects that a product have.

class ASPECTS_CORRELATION: ooObj
{
    private:
        char the_aspect_corr[20];
        float aspect_weightage_corrValue;

    public:
        ooHandle(ASPECTS_ar0) oneaspect1 <->
            someaspect_corr1[];
        ooHandle(ASPECTS_ar0) oneaspect2 <->
            someaspect_corr2[];
        ooHandle(ASPECTS_ar0) oneaspect3 <->
            someaspect_corr3[];
        ASPECTS_CORRELATION(char* aspect_corrname);
        void put_aspect_corr(char* aspect_corr);
        char* get_aspect_corr(void);
        float get_AspectcorrValue(void);
        void put_AspectcorrValue(float the_value);
        void readin(void);
        void readinnone(void);
        ooHandle(ASPECTS_ar0) get_aspect1(char* aspect_name1);
        void add_another_aspect1(ooHandle(ASPECTS_ar0) associated_aspect1);
        ooHandle(ASPECTS_ar0) get_aspect2(char* aspect_name2);
        void add_another_aspect2(ooHandle(ASPECTS_ar0) associated_aspect2);
        ooHandle(ASPECTS_ar0) get_aspect3(char* aspect_name3);
        void add_another_aspect3(ooHandle(ASPECTS_ar0) associated_aspect3);

};
Appendix I

// The code is written by A.R. Omar
// Loughborough University
//
// Last updates 7-6-96
// The benchmark.ddl file
// The objective is to get the comparison between the
// product and its competitor
//
// The COMPETITOR class primary function is to store the
// competitor have about a particular product.

class COMPETITOR: ooObj
{
    private:
        char competitor_name[50];

    public
        ooHandle(COMPETITOR_ASPECTVALUE) theASPECT_Value[]<-> thiscompetitor;
        ooHandle(COMPETITOR_primary_needScore)
            thePrNEED_Score[]<-> a_one_competitor;
        ooHandle(COMPETITOR_needScore) theNEED_Score[]<-> onecompetitor;
        ooHandle(COMPETITOR_SPECVALUE) theSPEC_Value[]<-> thecompetitor;
        COMPETITOR(char* competitorname);
        char* get_competitor(void);
        void put_competitor(char* itsname);
        void readin(void);
        ooHandle(COMPETITOR_SPECVALUE) get_specvalue(char* spec_value);
        void add_another_specvalue
            (ooHandle(COMPETITOR_SPECVALUE) assoc_specvalue);
        ooHandle(COMPETITOR_ASPECTVALUE) get_aspectvalue(char* aspect_value);
        void add_another_aspectvalue
            (ooHandle(COMPETITOR_ASPECTVALUE) assoc_aspectvalue);
        ooHandle(COMPETITOR_needScore) get_needscore(char* needscore);
        void add_another_needscore
            (ooHandle(COMPETITOR_needScore) assoc_needscore);
        int check_competitor_assoc
            (ooHandle(COMPETITOR_needScore)
                associated_compNeed_score);
        int check_compPr_assoc
            (ooHandle(COMPETITOR_primary_needScore)
                assoc_compPrNeed_score);
        int check_compSpecValue_assoc
            (ooHandle(COMPETITOR_SPECVALUE)
                associated_compSpec_value);

...benchmark.ddl
Appendix I

```cpp
int check_compAspectValue_assoc
    (ooHandle(COMPETITOR_ASPECTVALUE)
     assoc_compAspect_value);

// The COMPETITOR_needScore class primary function is to
// store the relationship value about customer need the
// competitor have

class COMPETITOR_needScore: ooObj
{
    private:
        int the_needScore;

    public:
        ooHandle(NEEDS_aRo) oneneed <- theCOM_SPECSCORE[];
        ooHandle(COMPETITOR) onecompetitor<-> theNEED_Score[];
        COMPETITOR_needScore(char* theneedScore);
        void put_need_score(int needScore_value);
        int get_need_score(void);
        void readin();
        ooHandle(NEEDS_aRo) get_needScore(char* needScore_name);
        void add_another_needScore(ooHandle(NEEDS_aRo)
                                    associated_needScore);
        ooHandle(COMPETITOR) get_competitor(char* competitor_name);
        void add_another_competitor(ooHandle(COMPETITOR)
                                     assoc_competitor);
};

// The COMPETITOR_ASPECTVALUE class primary function is to
// store the relationship value about COMPETITOR_ASPECT/
// VALUE the competitor have.

class COMPETITOR_ASPECTVALUE: ooObj
{
    private:
        float theCompetitor_aspectValue;

    public:
        ooHandle(COMPETITOR) thiscompetitor <->
            theASPECT_Value[];
        ooHandle(ASPECTS_aRo) thisaspect<->someaspect_value[];
        COMPETITOR_ASPECTVALUE(char* aspectValue_name);
        void put_competitor_aspectValue(float aspectValueValue)
            float get_aspectValue(void);
```

222 ...benchmark.ddl
void readin(void);
 ooHandle(COMPETITOR) get_competitor(char* competitor_name);
 void add_another_competitor(ooHandle(COMPETITOR) assoc_competitor);
 ooHandle(ASPECTS_aro) get_aspect(char* aspect_name);
 void add_another_aspect(ooHandle(ASPECTS_aro) assoc_aspect);

};

// The COMPETITOR_SPECVALUE class primary function is to store the relationship value about COMPETITOR_SPECVALUES the competitor have.

class COMPETITOR_SPECVALUE: ooObj {
  private:
    int theCompetitor_specValue;

  public:
    ooHandle(SPECIFICATION_aro) thisspec <-> someSpec_value[];
    ooHandle(COMPETITOR) thecompetitor <-> theSPEC_Value[];
    COMPETITOR_SPECVALUE(char* specValue_name);
    void put_competitor_specValue(int specValue);
    int get_specValue(void);
    void readin(void);
    ooHandle(COMPETITOR) get_competitor(char* competitor_name);
    void add_another_competitor(ooHandle(COMPETITOR) assoc_competitor);
    ooHandle(SPECIFICATION_aro) get_spec(char* spec_name);
    void add_another_spec(ooHandle(SPECIFICATION_aro) assoc_spec);

};
Appendix I

// The code is written by A.R.Omar
// Rev 1.0 dated 12-5-97
// The primary_need.ddl file
// The objective is to convert store customers' primary
// needs.

class primary_needs: public ooObj
{
private:
    char    the_primary_need[100];
    float   primary_needs_impt;
    float   primary_needspec_relativeImpt;
    float   primary_needsaspect_relativeImpt;
    float   primary_needs_Scorecompetitor;
    ooHandle(product_aro) a_prod <-> thePrimaryNEEDS[];

public:
    primary_needs(char* thename); // Construction by hand
    ooHandle(NEEDS_aro) thePrimary_needs[] <-> a_one_need;
    ooHandle(primary_needs_aspect_interRelation)
        primaryneed_INTER_aspect[] <-> need_aspect;
    ooHandle(primary_needspec_interRelation)
        primaryneed_INTER_spec[] <-> need_spec;
    ooHandle(COMPETITOR-primary_needScore)
        theCOM_PrNeed_SPECSCORE[] <-> a_one_need;
    ooHandle(VALUE_COEFF) pr_needvalue_coeff[] <-> a_pr_need;
    char* get_primary_needs(void);
    void put_primary_needs(char* primary_needs);
    float get_primary_needs_impt(void);
    void put_primary_needs_impt(float impt_rating);
    float getprimary_needspec_relativeImpt(void);
    void putprimary_needspec_relativeImpt(float specrelativeImpt_sum);
    float getprimary_needsaspect_relativeImpt(void);
    void putprimary_needsaspect_relativeImpt(float aspectrelativeImpt_sum);
    float getprimary_needs_Scorecompetitor(void);
    void put_primary_needs_Scorecompetitor(int Scorecompetitor);
    void readin(void);
    ooHandle(product_aro) get_product(char* product_name);
    void add_another_product(ooHandle(product_aro)
        associated_product);
    ooHandle(NEEDS_aro) get_primary_needs(char* primaryneeds_name);
    void add_another_primary_needs(ooHandle(NEEDS_aro)
        associated_primaryneeds);
Appendix I

```cpp
ooHandle(primary_needs_aspect_interRelation)
get_primary_needs_aspect_interRelation (char* aspects_name);
void add_another_primary_needs_aspect_interRelation
(ooHandle(primary_needs_aspect_interRelation)
associated_aspects);
ooHandle(primary_needs_spec_interRelation)
get_primary_needs_spec_interRelation(char* specs_name);
void add_another_primary_needs_spec_interRelation
(ooHandle(primary_needs_spec_interRelation)
associated_specs);
ooHandle(COMPETITOR_primary_needScore)
get_COMPETITOR_primary_needScore(char* needScore_name);
void add_another_COMPETITOR_primary_needScore
(ooHandle(COMPETITOR_primary_needScore)
associated_needScore);
int check_need_assoc(ooHandle(NEEDS_aro)
associated_need);
int check_pr_iaspect_assoc
(ooHandle(primary_needs_aspect_interRelation)
assoc_pr_iaspect);

// The primary_needs_aspect_interRelation class primary
// function is to store the relationship between customers'
// primary needs and aspect that a product have.

class primary_needs_aspect_interRelation: ooObj
{
private:
  char the_primaryneed_aspect[20];
  float primaryneed_aspect_Value;

public:
  ooHandle(primary_needs) need_aspect <->
  primaryneed_INTER_aspect[];
  ooHandle(ASPECTS_aro) aspect_one <->
  anASPECT_INTER[];
  primary_needs_aspect_interRelation(char* thename);
  void put_primary_needs_aspect_interRelation(char* aspect_inter);
  char* get_primary_needs_aspect_interRelation(void);
  float get_primaryneed_aspect_Value(void);
  void put_primaryneed_aspect_Value(float the_value);
  void readin(void);
```

...primary_needs.ddl
Appendix I

Appendix I

fooHandle(primary_needs) get_need(char* need_name);
void add_another_need(fooHandle(primary_needs)
    associated_need);
fooHandle(ASPECTS_ar) get_aspect(char* aspect_name);
void add_another_aspect(fooHandle(ASPECTS_ar)
    associated_aspect);
fooHandle(ASPECTS_ar) get_aspect_object();

class primary_needs_spec_interRelation: ooObj {
    private:
        char the_primaryneed_spec[20];
        int primaryneed_spec_Value;
        ooHandle(primary_needs) need_spec <->
        primaryneed_INTER_spec[];
        ooHandle(SPECIFICATION_ar) spec_one <->
        aSPEC_INTER[];

    public:
        primary_needs_spec_interRelation(char* thename);
        void put_primary_needs_spec_interRelation(char* spec_inter);
        char* get_primary_needs_spec_interRelation(void);
        int get_primaryneed_spec_Value(void);
        void put_primaryneed_spec_Value(int the_value);
        void readin(void);
        ooHandle(primary_needs) get_need(char* need_name);
        void add_another_need(ooHandle(primary_needs)
            associated_need);
        ooHandle(SPECIFICATION_ar) get_spec();
        void add_another_spec(ooHandle(SPECIFICATION_ar)
            associated_spec);
};

// The COMPETITOR_primary_needScore class primary function
// is to store the relationship value about customer

...primary_needs.ddl
Appendix I

// primary need that the competitor have.

class COMPETITOR_primary_needScore: ooObj
{
    private:
        float the_primary_needScore;
        ooHandle(primary_needs) a_one_need <->
            theCOM_PrNeed_SPECSCORE[];
        ooHandle(COMPETITOR) a_one_competitor<->
            thePrNEED_Score[];

    public:
        COMPETITOR_primary_needScore(char* theneedScore);
        void put_the_primary_needScore(float needScore_value);
        float get_the_primary_needScore(void);
        void readin();
        ooHandle(primary_needs) get_needScore(char*
            needScore_name);
        void add_another_needScore(ooHandle(primary_needs)
            associated_needScore);
        ooHandle(COMPETITOR) get_competitor(char*
            competitor_name);
        void add_another_competitor(ooHandle(COMPETITOR)
            assoc_competitor);
};
Appendix I

// Codes written by A.R. Omar modified from codes written
// by J A Harding, Dept Manufacturing Engineering, LUT.
// MOSES PROJECT - October 1994
//
// MOSES RULES SCHEMA
// rules.ddl
//
// Last changed 1/2/95

#include <working_memory.h>

enum logical { True, False, Unknown};

enum logical_comparator {less_than, less_than_or_equal,
greater_than,
greater_than_or_equal, equal, none};

enum logical_operator {AND, OR};

enum proceed_type {cont, stop};

enum value_type {float_val, integer_val};

enum memory_vars {sft, sft_ext, m_sft_in, m_sft_out,
sft_ext_in, sft_ext_out,
bear1, bear2};

enum memory_slots {ms_shaft, ms_shaft_ext, ms_bearing_A,
ms_bearing_B,
ms_reserve_factor, ms_uts, ms_uss, ms_full_load_power,
ms_full_load_speed,
ms_percent_start_torque, ms_calculation};

enum field_names {fn_name, fn_tappered, fn_normal,
fn_min_rad, fn_input_sec,
fn_output_sec, fn_input_loc, fn_input_hold, fn_output_loc,
fn_output_hold,
fn_cyl_sec_A, fn_cyl_sec_B, fn_groove_A, fn_groove_B,
fn_keyway_A, fn_keyway_B};

enum feature_type {cyl_shaft, tap_shaft, trans_null,
trans_rad, trans_u_rad,
trans_cham, trans_step, term_rad, term_cham, bl_r_h,
flat_b_r_h, slt, kway,
open_kway, grove};

enum creat_type {comp, feat};

// RULE_SET Class

...rules.ddl
Appendix I

class RULE_SET : public ooObj
{
    // number of rule_set_elements currently associated with
    // rule set
    private:
        char description[100];
        ooHandle(RULE_SET_ELEMENT) the_rules[] <->
            the_rule_set : prop(delete);
        char terminate_ins[100];
        int rule_set_number;
        int number_of_elements;
        int hasfired;

    public:
        RULE_SET();
        RULE_SET(char* thename);
        int fire_first(ooHandle(WORKING_MEMORY) my_memory);
        int fire_all(ooHandle(WORKING_MEMORY) my_memory);
        int fire_all_whilst(ooHandle(WORKING_MEMORY) my_memory,
            ooHandle(CONDITION) conH);
        char* get_termination_instructions();
        int get_rule_set_number();
        void print_description();
        void add_predefined_rule_to_set();
        void add_new_rule_to_set();
};

// RULE_SET_ELEMENT Class
class RULE_SET_ELEMENT : public ooObj
{
    private:
        int element_number;
        ooHandle(RULE) rule_element;
        ooHandle(RULE_SET) the_rule_set <->
            the_rules[] : prop(delete);

    public:
        RULE_SET_ELEMENT(char* thename, int number,
            ooHandle(RULE) aruleH);
        void re_number_element(int number);
        int get_element_number();
        ooHandle(RULE) get_rule();
};

// RULE Class
class RULE : public ooObj
Appendix I

```cpp
private:
    char ooHandle(CONDITION) : prop(delete);
    ooHandle(RESULTING_ACTION) : prop(delete);
    ooHandle(RULE_SET_ELEMENT) char int

public:
    RULE();
    RULE(char* thename);
    int fire(ooHandle(WORKING_MEMORY) my_memory, char* instructions);
    // Returns 1 if action carried out and 0 if rule does not fire
    void print_description();
    void initialise_result_message();

// CONDITION Class
class CONDITION : public ooObj {

    protected:
        char description[100];
        the_condition <-> the_rule
        the_result <-> the_rule
        rule_element <-> the_rule;
        completion_ins[100];
        hasfired;

    public:
        virtual void print();
        virtual int get_condition_value
            (ooHandle(WORKING_MEMORY) my_memory);

        // Returns 1 for True, 0 for False, & -1 for Error
        void readin_description();
        void print_description();
        void print_negate();
        logical get_negate();

// SIMPLE_CONDITION Class
class SIMPLE_CONDITION : public CONDITION {

};
```
private:
  ooHandle(EXPRESSION) the_element <->
  the_simp_con : prop(delete);

public:
SIMPLE_CONDITION();
SIMPLE_CONDITION(char* thename);
int get_condition_value(ooHandle(WORKING_MEMORY)
  my_memory);
// Returns 1 for True, 0 for False, & -1 for Error

// EXPRESSION Class
class EXPRESSION : public ooObj
{
  protected:
    ooHandle(SIMPLE_CONDITION) the_simp_con<->the_element :
    prop(delete);

  public:
// NOTE THERE IS NO CONSTRUCTOR HERE AS THIS IS
// INTENDED AS A SUPERCLASS FOR ALL THE DIFFERENT
// EXPRESSIONS
    virtual int get_expression_value
      (ooHandle(WORKING_MEMORY) my_memory);
// Returns 1 for True, 0 for False, & -1 for Error
};

// COMPOUND_CONDITION Class
class COMPOUND_CONDITION : public CONDITION
{
  private:
    ooHandle(SIMPLE_CONDITION) first_element :
    prop(delete);
    logical_operator conjunction;
    ooHandle(CONDITION) second_element :
    prop(delete);

  public:
COMPOUND_CONDITION();
COMPOUND_CONDITION(char* thename);
int get_condition_value(ooHandle(WORKING_MEMORY)
  my_memory);
// Returns 1 for True, 0 for False, & -1 for Error
};

// RESULTING_ACTION Class
Appendix I

class RESULTING_ACTION : public ooObj
{
    protected:
        ooHandle(RULE) the_rule <-> the_result : prop(delete);
        char result_message[100];

    public:
        // NOTE THERE IS NO CONSTRUCTOR HERE AS THIS IS INTENDED AS
        // AN ABSTRACT SUPERCLASS FOR SIMPLE_RESULTING_ACTION AND
        // COMPOUND_RESULTING_ACTION
        virtual char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
        void initialise_result_message(char* the_message);
    };

    // SIMPLE_RESULTING_ACTION Class
    class SIMPLE_RESULTING_ACTION : public RESULTING_ACTION
    {
        public:
            // NOTE THERE IS NO CONSTRUCTOR HERE AS THIS IS INTENDED AS
            // AN ABSTRACT SUPERCLASS FOR ALL THE ACTUAL CLASSES OF
            // RESULTING_ACTION
            char* execute_action(ooHandle(WORKING_MEMORY)
                my_memory);
    };

    // COMPOUND_RESULTING_ACTION Class
    class COMPOUND_RESULTING_ACTION : public RESULTING_ACTION
    {
        private:
            ooHandle(SIMPLE_RESULTING_ACTION) first_element:
                prop(delete);
            ooHandle(RESULTING_ACTION) second_element:
                prop(delete);
        public:
            COMPOUND_RESULTING_ACTION();
            COMPOUND_RESULTING_ACTION(char* thename);
            char* execute_action(ooHandle(WORKING_MEMORY)
                my_memory);
    };

    // USER_INPUT_RESPONSE_EXPRESSION Class
    // The primary function of objects of this class is to
    // write a pre-defined question to the screen, and obtain
    // the user's response. The message is requested when an
    // object of this class is created, and is held in the
Appendix I

// question attribute. If the user types 'y' or 'Y' in
// response to the question, this expression returns 1
// (TRUE), if the user types any other
// character, this expression returns 0 (FALSE).
class USER_INPUT_RESPONSE_EXPRESSION : public EXPRESSION {
    private:
        char question[200];
    public:
        USER_INPUT_RESPONSE_EXPRESSION(char* thename);
        int get_expression_value(ooHandle(WORKING_MEMORY)
            my_memory);
    // Returns 1 for True, 0 for False, & -1 for Error

    // ALWAYS_TRUE_EXPRESSION Class
    // The primary function of objects of this class is to
    // ensure that the its related resulting action is ALWAYS
    // carried. ie This expression ALWAYS returns 1 (TRUE).
class ALWAYS_TRUE_EXPRESSION : public EXPRESSION {
    public:
        ALWAYS_TRUE_EXPRESSION(char* thename);
        int get_expression_value(ooHandle(WORKING_MEMORY)
            my_memory);
    // Returns 1 for True, 0 for False, & -1 for Error

    // MENU_SELECTION_MADE_EXPRESSION Class
    // The primary function of objects of this class is to
    // ensure that a valid selection is made from a menu.
    // There can be a maximum of 20 elements in the menu,
    // currently. If one of the menu list elements is chosen,
    // its number will be entered into the TEMP VALS integer
    // value slot of the working memory object, my_memory, and
    // this expression will return 1 (TRUE).
    // If the value 0 is entered, ie the select nothing from
    // the menu option, this expression will return 0 (FALSE).
    // (If the size of the array, menu_menu_elements is
    // increased above 20, the size of the rulenames_list array
    // in class FIRE_A_SELECTED_RULE_ACTION can also be
    // increased).
class MENU_SELECTION_MADE_EXPRESSION : public EXPRESSION {
    private:
        char menu_header[100];
        char menu_elements[20][100];
        char menu_comment[100];
        int number_of_menu_elements;
public:
  MENU_SELECTION_MADE_EXPRESSION(char* thename);
  int get_expression_value(ooHandle(WORKING_MEMORY) my_memory);
    // Returns 1 for True, 0 for False, & -1 for Error
};

// WI_MENU_SELECTION_MADE_EXPRESSION Class
// The primary function of objects of this class is to
// ensure that a valid selection is made from a menu.
// There can be a maximum of 20 elements in the menu,
// currently. If one of the menu list elements is chosen,
// its number will be entered into the TEMP VALS integer
// value slot of the working memory object, my_memory, and
// this expression will return 1 (TRUE).
// If the value 0 is entered, i.e. the select nothing from
// the menu option, this expression will return 0 (FALSE).
// (If the size of the array, menu_menu_elements is
// increased above 20, the size of the rulenames_list array
// in class FIRE_A_SELECTED_RULE_ACTION can also be
// increased).
class WI_MENU_SELECTION_MADE_EXPRESSION : public EXPRESSION {
private:
  char menu_header[100];
  char menu_elements[20][100];
  char menu_comment[100];
  int number_of_menu_elements;

public:
  WI_MENU_SELECTION_MADE_EXPRESSION(char* thename);
  int get_expression_value(ooHandle(WORKING_MEMORY) my_memory);
    // Returns 1 for True, 0 for False, & -1 for Error
};

// PRINT_MESSAGE_ACTION Class
// The primary function of objects of this class is to
// write a pre-defined message to the screen. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class PRINT_MESSAGE_ACTION : public SIMPLE_RESULTING_ACTION {
private:
  char message[200];

public:
  PRINT_MESSAGE_ACTION(char* thename);
  char* execute_action(ooHandle(WORKING_MEMORY) my_memory);
};
Appendix I

// POSTSCRIPT_DISPLAY_ACTION Class
// The primary function of objects of this class is to
display a postscript file, whose name is stored in the
message attribute of objects of this class, and which
must be stored in the Postscript_files directory. The
postscript file is displayed using Pageview, on the
workstation specified by the value of the machine_name
attribute in the working memory object
(my_memory), which must be passed as a parameter to the
execute_action method of objects of this class.
Class POSTSCRIPT_DISPLAY_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
char message[100]; // postscript file name
char machine_name[20];
// name of machine image is displayed on
public:
POSTSCRIPT_DISPLAY_ACTION(char* thename);
char* execute_action(ooHandle(WORKING_MEMORY)
    my_memory);
};

// READ_INTO_MEMORY_ACTION Class
// The primary function of objects of this class is to
update a slot in the TEMP_VALUES object in the Working
Memory of an expert with a value provided by the user at
run time.
class READ_INTO_MEMORY_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
char prompt_message[200];
char var_is_a[20];
public:
READ_INTO_MEMORY_ACTION(char* thename);
char* execute_action(ooHandle(WORKING_MEMORY)
    my_memory);
void print(void);
};

// FIRE_A_SELECTED_RULE_ACTION Class
// The primary function of objects of this class is to
select a rule to fire, dependent upon the current value
of the integer slot in the TEMP_VALUES object. The names
of the rules which might possibly be fired are stored
in the attribute rulename_list. So currently, objects of
this class can select from a maximum of 20 possible
rules which might be fired. This number could be
Appendix I

// increased if the size of the menu_elements array
// attribute of MENU_SELECTION_MADE_EXPRESSION class is
// increased. The number of rules which can be selected
// from is stored in the attribute, num_of_rules, and
// this, and the rule names, must be specified when
// initialising objects of this class.

class FIRE_A_SELECTED_RULE_ACTION : public
  SIMPLE_RESULTING_ACTION
{
  private:
    char           rulename_list[20][100];
    int            num_of_rules;

  public:
    FIRE_A_SELECTED_RULE_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
                         my_memory);
    void print(void);
};
Appendix I

// Written by J.A. Harding, Dept Manufacturing Eng., LU.
// MOSES PROJECT - October 1994
/
// MOSES WORKING MEMORY SCHEMA
// working_memory.ddl
/
// The revised code is written by AR OMAR,
// Manufacturing Dept, Loughborough Univ
// as originally written by Jenny Harding
// Last Changed 3/2/97
/
/
#include <needs_spec.h>
#include <inter_correlate.h>
#include <benchmark.h>
#include <primary_need.h>
/
// TEMP_VALUES Class
class TEMP_VALUES : public ooObj
{
    protected:
    astring[100];
    afloat;
    anint;
    ooHandle(HOQ_W_M) wmhoql <-> temp_vals : prop(delete);

    public:
    TEMP_VALUES();
    void print();
    void win_print();
    void update_string(char* slot, char* update_val);
    void update_float(char* slot, float update_val);
    void update_int(char* slot, int update_val);
    char* get_string(char* slot);
    float get_float(char* slot);
    int get_int(char* slot);
};
/
// WORKING_MEMORY Class
class WORKING_MEMORY : public ooObj
{
    public:
    // There is no CONSTRUCTOR for this class as it is an
    // abstract super class and therefore should never be
    // created
    virtual void print();

    ...working_memory.ddl
virtual void update_string(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib, char* update_val);
virtual void update_float(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib, float update_val);
virtual void update_int(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib, int update_val);
virtual char* get_string(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib);
virtual float get_float(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib);
virtual int get_int(char* slot, char* slot_attrib, char* slot_attrib_slot, char* slot_attrib_slot_attrib);
};

class HOQ_W_M : public WORKING_MEMORY
{
protected:
    char     DB_Name[100];  // Database for output
    char     VOC_Name[100];
    char     Comp_Name[100]; // Name of current Voice of Customer

public:
    ooHandle(VALUE_COEFF)value[] <-> wm_value :
        prop(delete);
    ooHandle(VALUE_COEFF)current_value <-> wm_cvalue :
        prop(delete);
    ooHandle(TEMP_VALUES)  temp_vals <-> wmhoql :
        prop(delete);
    HOQ_W_M(char* thename);
    void put_DB_Name(char* db_name);
    char* get_DB_Name(void);
    void initialise_memory();
    void add_another_V_coeff(ooHandle(VALUE_COEFF) associated_vcoeff);
    void print();

};

// The value_coefficient class
Appendix I

class VALUE_COEFF : public ooObj
{
private:
  float  benefit_value;

public:
  ooHandle(HOQ_W_M)  wm_value  <->  value[]  :  prop(delete);
  ooHandle(HOQ_W_M)  wm_cvalue  <->  current_value  :
                 prop(delete);
  ooHandle(primary_needs)  a_pr_need  <->
        pr_needvalue_coeff[]  :  prop(delete);
  ooHandle(NEEDS_aro)   a_need  <->  needvalue_coeff[]
               :  prop(delete);
  ooHandle(ASPECTS_aro) an_aspect  <->  aspectvalue_coeff[]
               :  prop(delete);
  ooHandle(SPECIFICATION_aro)  a_spec  <->
     specvalue_coeff[]  :  prop(delete);
  VALUE_COEFF(char *  thename);
  void print();
  void create_coeff_value();
  void putVcoeff(float value);
  float getVcoeff(void);
  ooHandle(primary_needs)  get_pr_need();
  ooHandle(SPECIFICATION_aro)  get_spec();
  ooHandle(NEEDS_aro)  get_need();
  ooHandle(ASPECTS_aro)  get_aspect();
  int get_thespecValue(ooHandle(COMPETITOR)  own_product);
  void add_another_hoq(ooHandle(HOQ_W_M)  associated_hoq);
  void add_another_needs(ooHandle(NEEDS_aro)  associated_needs);
  void add_another_spec(ooHandle(SPECIFICATION_aro)
           associated_spec);
  void add_another_aspects(ooHandle(ASPECTS_aro)
           associated_aspects);
  void add_another_pr_need(ooHandle(primary_needs)
      assoc_pr_need);
  void add_another_specValue
     (ooHandle(COMPETITOR_SPECVALUE)  assoc_specValue);
  int  check_need_assoc(ooHandle(NEEDS_aro)  assoc_need);
  int  check_spec_assoc(ooHandle(SPECIFICATION_aro)
      assoc_spec);
  int  check_aspect_assoc(ooHandle(ASPECTS_aro)
      assoc_aspect);
  int  check_pr_need_assoc(ooHandle(primary_needs)
      assoc_pr_need);
};
Appendix I

// Written by J A Harding, Dept Manufacturing Eng., LU.
// MOSES PROJECT - February 1995
//
// MOSES RULES SCHEMA
// mod_rules.ddl
// (Moderator specific rules)
// The revised code is written by AR OMAR,
// Manufacturing Dept, Loughborough Univ
// as originally written by Jenny Harding
// Last changed 13/4/95

#include <working_memory.h>
#include <rules.h>

// WI_USER_INPUT_RESPONSE_EXPRESSION Class
// The primary function of objects of this class is to
// write a pre-defined question to the screen, and obtain
// the user's response. The message is requested when an
// object of this class is created, and is held in the
// question attribute. If the user types 'y' or 'Y' in
// response to the question, this expression returns 1
// (TRUE), if the user types any other character, this
// expression returns 0 (FALSE).
class WI_USER_INPUT_RESPONSE_EXPRESSION : public EXPRESSION
{
    private:
        char question[200];

    public:
        WI_USER_INPUT_RESPONSE_EXPRESSION(char* thename);
        int get_expression_value(ooHandle(WORKING_MEMORY) my_memory);
        // Returns 1 for True, 0 for False, & -1 for Error
};

// WI_PRINT_MESSAGE_ACTION Class
// The primary function of objects of this class is to
// write a pre-defined message to the screen. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_MESSAGE_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];

    public:
        WI_PRINT_MESSAGE_ACTION(char* thename);
Appendix I

```
char* execute_action(ooHandle(WORKING_MEMORY)
    my_memory);
);

// WI_READ_INTO_MEMORY_ACTION Class
// The primary function of objects of this class is to
// update a slot in the TEMP_VALUES object in the Working
// Memory of an expert with a value provided by the user at
// run time.
class WI_READ_INTO_MEMORY_ACTION : public
SIMPLE_RESULTING_ACTION
{
    private:
        char prompt_message[200];
        char var_is_a[20];

    public:
        WI_READ_INTO_MEMORY_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
        void print(void);
    };

// SET_INITIAL_BENEFIT_VALUES_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class SET_INITIAL_BENEFIT_VALUES_ACTION : public
SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];

    public:
        SET_INITIAL_BENEFIT_VALUES_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
    };

// WI_PRINT_BENEFIT_VALUES_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_BENEFIT_VALUES_ACTION : public
SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];
```
public:

WI_PRINT_BENEFIT_VALUES_ACTION(char* thename);
char* execute_action(ooHandle(WORKING_MEMORY)
               my_memory);

};

// CHECK_BENEFIT_VALUES_SPECBENCHMARK_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class CHECK_BENEFIT_VALUES_SPECBENCHMARK_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
    char message[200];

public:
    CHECK_BENEFIT_VALUES_SPECBENCHMARK_ACTION(char* 
             thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
               my_memory);

};

// CHECK_BENEFIT_VALUES_RELATION_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class CHECK_BENEFIT_VALUES_RELATION_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
    char message[200];

public:
    CHECK_BENEFIT_VALUES_RELATION_ACTION(char* 
                thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
               my_memory);

};

// WI_PRINT_TOTAL_BENEFIT_VALUES_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_TOTAL_BENEFIT_VALUES_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
    char message[200];


Appendix I

```cpp
public:
    WI_PRINT_TOTAL_BENEFIT_VALUES_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
        my_memory);
};

// WI_PRINT_SORT_BENEFIT_VALUES_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_SORT_BENEFIT_VALUES_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char      message[200];

    public:
        WI_PRINT_SORT_BENEFIT_VALUES_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
    };

// WI_COMPUTE_ASPECT_MEAN_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_COMPUTE_ASPECT_MEAN_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char      message[200];

    public:
        WI_COMPUTE_ASPECT_MEAN_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
    };

// WI_COMPUTE_COMSPEC_MEAN_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_COMPUTE_COMSPEC_MEAN_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char      message[200];

    public:
```
Appendix I

WI_COMPUTE_COMSPEC_MEAN_ACTION(char* thename);
char* execute_action(ooHandle(WORKING_MEMORY)
    my_memory);
}

// CHECK_BV_RELATION_ASPECT_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class CHECK_BV_RELATION_ASPECT_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];

    public:
        CHECK_BV_RELATION_ASPECT_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
    }

// CHECK_BV_ASPECTBENCHMARK_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class CHECK_BV_ASPECTBENCHMARK_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];

    public:
        CHECK_BV_ASPECTBENCHMARK_ACTION(char* thename);
        char* execute_action(ooHandle(WORKING_MEMORY)
            my_memory);
    }

// WI_PRINT_TOTAL_BV_ASPECT_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_TOTAL_BV_ASPECT_ACTION : public
    SIMPLE_RESULTING_ACTION
{
    private:
        char message[200];

    public:
        WI_PRINT_TOTAL_BV_ASPECT_ACTION(char* thename);
Appendix I

```c
char* execute_action(ooHandle(WORKING_MEMORY)
  my_memory);
}

// WI_PRINT_SORT_BV_ASPECT_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_SORT_BV_ASPECT_ACTION : public
  SIMPLE_RESULTING_ACTION
{
  private:
    char               message[200];

  public:
    WI_PRINT_SORT_BV_ASPECT_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
      my_memory);
};

// WI_PRINT_BV_SpVsAs_ACTION Class
// The primary function of objects of this class is to
// create update value of VALUE_COEFF object. The message
// is requested when an object of this class is created,
// and is held in the message attribute.
class WI_PRINT_BV_SpVsAs_ACTION : public
  SIMPLE_RESULTING_ACTION
{
  private:
    char               message[200];

  public:
    WI_PRINT_BV_SpVsAs_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
      my_memory);
};
```

...mod_rules.ddl
// Written by AR OMAR, Dept Manufacturing Eng., LU.
// MOSES RULES SCHEMA
// mod1_rules.ddl
// (Moderator specific rules)
// Last changed 16/10/97

#include <working_memory.h>
#include <rules.h>

// COMPUTE_NEED_IA_MEAN_ACTION Class
// The primary function of objects of this class is to
// calculate mean value of CUSTOMER_NEEDS. The message is
// requested when an object of this class is created, and
// is held in the message attribute.
class COMPUTE_NEED_IA_MEAN_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
    char         message[200];

public:
    COMPUTE_NEED_IA_MEAN_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
                         my_memory);
};

// COMPUTE_COMNEED_MEAN_ACTION Class
// The primary function of objects of this class is to
// calculate mean value of CUSTOMER_NEED BENCHMARK. The
// message is requested when an object of this class is
// created, and is held in the message attribute.
class COMPUTE_COMNEED_MEAN_ACTION : public
SIMPLE_RESULTING_ACTION
{
private:
    char         prompt_message[200];
    char         var_is_a[20];

public:
    COMPUTE_COMNEED_MEAN_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
                         my_memory);
    void print(void);
};

// SET_BV_PrNeed_ACTION Class
// The primary function of objects of this class is to set
// initial value which compare against Primary Needs. The
// message is requested when an object of this class is
// created, and is held in the message attribute.
Appendix I

class SET_BV_PrNeed_ACTION : public
  SIMPLE_RESULTING_ACTION
{
  private:
    char      message[200];

  public:
    SET_BV_PrNeed_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
      my_memory);
};

// CHECK_BV_PrNeed_RELATION_ACTION Class
// The primary function of objects of this class is to
// calculate value against interrelationship and
// correlation weightage value. The message is requested
// when an object of this class is created, and is held in
// the message attribute.
class CHECK_BV_PrNeed_RELATION_ACTION : public
  SIMPLJ_RESULTING_ACTION
{
  private:
    char    message[200];

  public:
    CHECK_BV_PrNeed_RELATION_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
      my_memory);
};

// WI_PRINT_TOTAL_BV_PrNeed_ACTION Class
// The primary function of objects of this class is to
// calculate total value of each BV. The message is
// requested when an object of this class is created, and
// is held in the message attribute.
class WI_PRINT_TOTAL_BV_PrNeed_ACTION : public
  SIMPLtResultadoN_ACTION
{
  private:
    char      message[200];

  public:
    WI_PRINT_TOTAL_BV_PrNeed_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY)
      my_memory);
};

// WI_PRINT_SORT_BV_PrNeed_ACTION Class
// The primary function of objects of this class is to sort
// value in order of priority. The message is requested
// when an object of this class is created, and is held in
// the message attribute.
class WI_PRINT_SORT_BV_PrNeed_ACTION : public SIMPLE_RESULTING_ACTION
{
private:
    char message[200];

public:
    WI_PRINT_SORT_BV_PrNeed_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY) my_memory);
};

// WI_PRINT_BV_PrVsAs_ACTION Class
// The primary function of objects of this class is to
// create display output. The message is requested when an
// object of this class is created, and is held in the
// message attribute.
class WI_PRINT_BV_PrVsAs_ACTION : public
    SIMPLE_RESULTING_ACTION
{
private:
    char message[200];

public:
    WI_PRINT_BV_PrVsAs_ACTION(char* thename);
    char* execute_action(ooHandle(WORKING_MEMORY) my_memory);
};
### Appendix II

**Lists of Knowledge Representation Model’s Rules**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET INITIAL BENEFIT VALUES ACTION</td>
<td>This type of rule check the status of each NEEDS object of own product against competitors’ benchmark value. If the value is greater than competitors’ need score value, there will be no change and each specification in this row of the HoQ is given a value of zero.</td>
</tr>
<tr>
<td>WI_PRINT BENEFIT VALUES ACTION</td>
<td>Prints an output display of coefficient values of customers’ secondary needs and design features’ specifications.</td>
</tr>
<tr>
<td>CHECK BENEFIT VALUES SPECBENCHMARK ACTION</td>
<td>Locates INTERRELATIONSHIP object and compare the specification’s coefficient value of own product against competitors’ benchmarks for that particular object. If the coefficient value for own product is greater, then the coefficient value for this specification will be zero.</td>
</tr>
<tr>
<td>CHECK BENEFIT VALUES RELATION ACTION</td>
<td>Correlates INTERRELATIONSHIP and CORRELATION specification objects under observation. Using a mathematical equation, it evaluates the relationship.</td>
</tr>
<tr>
<td>WI_PRINT TOTAL BENEFIT VALUES ACTION</td>
<td>Computes the summation of coefficient values for every specification.</td>
</tr>
<tr>
<td>WI_PRINT_SORT BENEFIT VALUES ACTION</td>
<td>Sorts coefficient values after it has been totalled and notifies designer which specifications require modification in order of priority.</td>
</tr>
<tr>
<td>WI_COMPUTE ASPECT MEAN ACTION</td>
<td>Computes mean value INTERRELATIONSHIP objects of secondary needs against specifications and assign this value as an attribute to ASPECT INTERRELATIONSHIP object.</td>
</tr>
<tr>
<td>WI_COMPUTE COMSPEC MEAN ACTION</td>
<td>Computes mean value of specification benchmark objects of each competitor and assign as an attribute to ASPECT benchmark object.</td>
</tr>
<tr>
<td>CHECK BV RELATION ASPECT ACTION</td>
<td>Correlates INTERRELATIONSHIP and CORRELATION aspect and secondary need objects under observation. Using a mathematical equation, it evaluates the relationship.</td>
</tr>
<tr>
<td>CHECK BV ASPECT BENCHMARK ACTION</td>
<td>Locates INTERRELATIONSHIP object and compare the aspect’s coefficient value of own product against competitors’ benchmarks for that particular object. If the coefficient value for own product is greater, then the coefficient value for this specification will be zero.</td>
</tr>
</tbody>
</table>
### Appendix II

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WI_PRINT_TOTAL_BV_ASPECT ACTION</td>
<td>Computes the summation of coefficient values for every aspect.</td>
</tr>
<tr>
<td>WI_PRINT_SORT_BV_ASPECT ACTION</td>
<td>Sorts coefficient values after it has been totalled and notifies designer which aspects require modification in order of priority.</td>
</tr>
<tr>
<td>WI_PRINT_BV_SpVsAs ACTION</td>
<td>Prints an output display of coefficient values of customers’ secondary needs and design features’ aspects</td>
</tr>
<tr>
<td>COMPUTE_NEED_IA_MEAN ACTION</td>
<td>Computes mean value of secondary need against aspect objects and assign as an attribute to INTERRELATION primary need against aspect object.</td>
</tr>
<tr>
<td>COMPUTE_COMNEED_MEAN ACTION</td>
<td>Computes mean value of secondary need benchmark objects of each competitor and assign as an attribute to primary need benchmark object.</td>
</tr>
<tr>
<td>SET_BV_PrNeed ACTION</td>
<td>This type of rule check the status of each Primary needs object of own product against competitors’ benchmark value. If the value is greater than competitors’ need score value, there will be no change and each specification in this row of the HoQ is given a value of zero.</td>
</tr>
<tr>
<td>CHECK_BV_PrNeed_RELATION ACTION</td>
<td>Correlates INTERRELATIONSHIP and CORRELATION aspect and primary need objects under observation. Using a mathematical equation, it evaluates the relationship.</td>
</tr>
<tr>
<td>WI_PRINT_TOTAL_BV_PrNeed ACTION</td>
<td>Computes the summation of coefficient values for every aspect that contemplate with primary need.</td>
</tr>
<tr>
<td>WI_PRINT_SORT_BV_PrNeed ACTION</td>
<td>Sorts coefficient values after it has been totalled and notifies designer which aspects that contemplate with primary needs require modification in order of priority.</td>
</tr>
<tr>
<td>WI_PRINT_BV_PrVsAs ACTION</td>
<td>Prints an output display of coefficient values of customers’ primary needs and design features’ aspects</td>
</tr>
</tbody>
</table>