

Ethias ✓

NOTTINGHAM
TRENT UNIVERSITY

**Libraries and Learning Resources
SHORT LOAN COLLECTION**

Date	Time	Date	Time
NTU 21 JUN 2006	Rf		
25 OCT 2006	10		
2055 pm			

Please return this item to the issuing library.
Fines are payable for late return.

THIS ITEM MAY NOT BE RENEWED

Short Loan 06

06 APR 2006

818=324926

40 0764909 2



ProQuest Number: 10290217

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10290217

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Web-based Collaborative Environment for Integrated Design

Shuyan Ji

A thesis submitted in partial fulfilment of the
requirements of Nottingham Trent University
for the degree of Doctor of Philosophy

This research programme was carried out at
Advanced Design and Manufacturing Engineering Centre,
School of Architecture, Design and Built Environment,
Nottingham Trent University,
Burton Street, Nottingham,
NG1 4BU, UK

March 2006

Abstract

The total product design process consists of several stages including the formulation of product design specifications, conceptual design, detail design, manufacture and sales. In recent years, the rapid development of computer technologies has greatly impacted the product development process. In the engineering area, powerful computer-based tools such as Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems enable engineers to fulfil various tasks in individual stages such as conceptual and detail design, and manufacturing simulation. However, the increasing complexity of modern products, cruel competition pressure and the globalisation of product development necessitate a collaborative design environment where distributed computer programs and dispersed experts in similar or different domains could be collaboratively involved in a common design activity, in order to obtain high quality and low cost product design. The key issues for establishing such a collaborative environment are the integration and communication of the computerised design resources with heterogeneity and distribution features.

The collaborative architecture presented in this thesis employs a combination of CORBA with other technologies such as Java, Genetic Algorithms to integrate engineering design programs and to enable communication between them, regardless of languages they are written in and platforms they are ported on, and to leverage multiple design interests. The resultant Web-based architectures have identified and addressed three main application paradigms. Firstly, by using CORBA-Servlet, a singular large-size program can be executed remotely with interactive features such as parameter input, program execution and monitoring, and varieties of dynamic and rich forms of resultant return. The developed system also facilitates multi-user management. Secondly, Genetic Algorithms has been employed to combine the CORBA to mediate the impacts from different domain expert considerations over the distributed environment. In this environment, CORBA is used as an architecture for integrating

multiple design applications that are heterogeneous and distributed and for enabling them to communicating with each other, a GA-based optimiser is designed to help a main designer to conduct gear design optimisations through invoking remote and heterogeneous applications to meet user's complex needs. Thirdly, Applet-CORBA based system is developed to provide a thin client model to enable users to do the remote invocation from a Web browser, without any setting up for CORBA in advance. As a user of this application, a designer does not need to get to know anything about CORBA. The rich features of applet allow client developers to design varieties forms of Web browser application, such as dynamic data, dynamic drawing, and so on.

It is found that functionalities of traditional, stand-alone, single-user computer-aided applications can be extended by employing modern distributed object computing and web technologies. These technologies provide cornerstone and effective support in building scalable, extensible, and interactive distributed systems for collaborative design, as illustrated in demonstration systems developed in the present thesis.

Keywords: Distributed System, Web-based Collaborative Environment, CORBA, Genetic Algorithms, Java, Servlet, Applet, Gear Design

Acknowledgements

The work presented in this thesis was accomplished under the supervision of Professor Daizhong Su, Professor John Leslie Henshall and Professor Barry Hull at Nottingham Trent University. I would like to thank my Director of Studies, my Supervisor, Professor Daizhong Su, for his continuous support, encouragement and guidance, and for having faith in me throughout this project. Sincere thanks are due to my supervisors: Professor Les Henshall and Professor Barry Hull for their advice, comments, backing and interest in my work. Their knowledge and experience have been invaluable in the development of this dissertation.

I would like to express my gratitude for all the help and advice given by the members of research administrators, technicians and fellow researchers. Although it is not possible to enumerate all concerned, I would like to especially thank Mrs Doreen Corlett, Ms Julie Bradshaw, Mr Gary Griffiths, my friends and colleagues, for their kindly help.

Finally, my greatest thanks to my family, Jiansheng, my husband, Mengqi, my daughter, for their unequivocal love and support.

Shuyan Ji
30/11/2005

Publications and Presentations

Book Chapter

- S. Ji and D. Su, Review of Web-based collaborative design, *Web-enabled Collaborative Design and Manufacture – Literature Review, Research and Development*, ed by Daizhong Su, pp129-143, 2005, ISBN 1-84233-114-0, EU Asia IT&C – WECIDM

Refereed Journal Papers

- S. Ji, D. Su and J. Li, Integration, management and communication of heterogeneous resources based on Web technologies, *Lecture Notes in Computer Science*, Volume 3865 / 2005, UK
- D. Su, J. Li and S. Ji, Online collaborative design within a Web-enabled environment, *Lecture Notes in Computer Science*, Volume 3168 / 2005, ISSN 0302-9743 Springer-Verlag GmbH.
- D. Su, S. Ji, N. Amin and J. B. Hull, 2003, Multi-user Internet environment for gear design optimisation, *Integrated Manufacturing Systems*, Vol. 14, No.6, 2003, pp 498-507, MCB UP Limited

Refereed International Conference Papers

- S. Ji, and D. Su, A heterogeneous collaborative design environment with dynamic management features, *Proceedings of the 9th International conference on Computer Cooperative Work in Design*, ed by W. Shen, 24-26 May, 2005, Coventry, UK, pp 690-695
- S. Ji, D. Su, J. L. Henshall and J. B. Hull, Gear design optimisation using a Genetic Simulated Annealing Algorithm, poster proceedings, *International Conference on Adaptive Computing in Design and Manufacture*, 20 - 22 April 2004, Bristol, UK, pp 5-9
- W. Peng, Y. Xiong, S. Ji and D. Su, A virtual research institute and its utilisation in CAE for worm gear drives and gear design optimisation, *8th*

International Conference on Computer Supported Cooperative Work in Design, 26-28 May 2004, Xiamen, China, pp 552-557

- S. Ji, J. Li. Internet-based design environment using CORBA and Java. *1st Annual Academic Conference of CSSA-Nottingham*, 8th July 2003, Nottingham, UK
- J. B. Hull, D. Su and S. Ji, 2003, Development of a powerful software tool for collaborative design and manufacture over the Internet, *Proceedings of the International Conference on Industrial Tools*, 8-12, April, 2003, Bled, Slovenia, pp 399-402
- D. Su, S. Ji, N. Amin and X Chen, 2002, A framework of Web support for collaborative design, *Proceedings of the 5th International Conference on Frontiers of Design and Manufacturing*, Volume 1, Dalian, China, pp 492-498
- D. Su, S. Ji, J. Li and J. B. Hull, 2002, Web-enabled Collaborative Environment for Integrated Design and Manufacture, *Proceedings of Concurrent Engineering*, 27-31 July, Cranfield, pp 93-101
- S. Ji, D. Su and J. B. Hull, 2002, Application of Java Servlet technique for Internet-based design optimisation, *Proceedings of 4th International Conference on Mechanics and Materials in Design*, 5-8 June, Nagoya, Japan, pp 198-200
- D. Su, S. Ji, N. Amin and J. B. Hull, 2001, An Internet-based system of gear design optimisation using Java Servlets, *Proceedings of The International Conference on Computer Aided Industrial Design and Conceptual Design*, 16-20 October 2001, Jinan, China, pp 30-36
- S. Ji, and D. Su, Integrated worm gearing design based on distributed heterogeneous system, *Proceedings of International Conference on Advanced Design and Manufacture (ADM2006)*, ed by D. Su and S. Zhu, 8-10 Jan. 2006, Harbin, China, pp 429-432
- S. Ji and D. Su, Gear design optimisation with a variable penalty function, *Proceedings of International Conference on Advanced Design and Manufacture (ADM2006)*, ed by D. Su and S. Zhu, 8-10 Jan. 2006, Harbin, China, pp 629-632

Notions

Symbol	Description
ϕ_d	Face width coefficient: the ratio of face width to pitch diameter of the pinion
m	Module
h_{ap}	Gear addendum coefficient
α	Pressure angle
β	Helical angle
ρ_{fp}	Rack tip radius coefficient: the product of ρ_{fp} and m is the rack tip radius
x_1	Pinion addendum coefficient
x_2	Wheel addendum coefficient
z_1	Pinion tooth number
a	Centre distance
u	Transmission ratio
σ_{b1}	Pinion bending stress
σ_{b2}	Wheel bending stress
ε	Contact ratio
σ_H	Contact stress
$[\sigma_H]$	Permission contact stress
$[\sigma_F]$	Permission bending stress
B	Gear face width
t	Generation
T	Temperature schedule parameter

Abbreviations

Abbreviation	Description
ACI	Advanced Collaborative Infrastructure
ANTS	Advanced .NET Testing System
ASP	Active Server Pages
CAD	Computer Aided Design
CAM	Computer Aided Manufacture
CAX	A summary term for various kinds of Computer Aided technologies
CGI	The Common Gateway Interface
CORBA	Common Object Request Broker Architecture
DCE	Distributed computing environment
DCOM	Distributed Component Object Model
DFM	Design for Manufacture
DFX	Design for X
DGDO	Distributed Gear Design Optimisation
FTP	File Transfer Protocol
GA	Genetic Algorithms
GUI.	Graphics User Interface
HTML	Hyper Text Markup Language
HTTP	Hypertext Transport Protocol
HTTPS	Secure Hypertext Transport Protocol
IDC	Industrial Development Corporation
IDL	The Interface Definition Language
IGES	Initial Graphics Exchange Specification
IIOP	The Internet Inter-ORB Protocol
IPC	Inter-process communication
JDBC	Java Database Connection
JESS	Java Expert System Shell

JMS	Java Message Service
JVM	Java virtual Machine
KQML	knowledge query manipulation Language
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
ORB	The Object Request Broker
PDM	Product Data Management
PDS	Product Design Specifications
RMI	Remote Method Invocation
RPC	Remote Procedure Calls
SGML	Standard Generalised Markup Language
SMTP	Simple Mail Transfer Protocol
SOAP	The Simple Object Access Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
VRML	Virtual Reality Modeling Language
W3C	The World Wide Web Consortium
WSDL	The Web Service Description Language
WS-I	Web Services Interoperability Organisation
WWW	World-Wide Web
XML	eXtensible Markup Language

Content

Abstract	I
Acknowledgements	III
Publications and Presentations	IV
Notions	VI
Abbreviations	VII
Content	IX
List of Figures	XVII
List of Tables	XX
Chapter 1 Introduction	1
1.1 Background	1
1.2 Collaborative Design Requirements	3
1.3 Aims, Objectives and Scope of the Research.....	6
1.4 Research Approach	7
1.5 Outline of This Thesis	8
Chapter 2 Review for Web-based Collaborative Design	11
2.1 Fundamentals of Collaborative Design.....	11
2.1.1 Design Process, Concurrent Engineering & Collaborative Engineering	11
2.1.2 Collaborative Design & the Internet	15
2.2 Web-based Collaborative Design.....	17
2.3 Data Exchange	24
2.4 Genetic Algorithms	27
2.5 Other Web-based Technologies	27
2.6 Gear Design.....	28
2.7 Concluding Remarks	29
Chapter 3 Overview on Architectures of Distributed System.....	32
3.1 Introduction.....	32
3.2 Fundamental Architectures of Distributed System	33

3.3 Point-to-Point Communication Structure.....	35
3.3.1 CORBA-Based Point-to-Point Structure	35
3.3.2 Comparison of CORBA and Other Distributed Technologies.....	38
3.4 CORBA-Based Server-Centralised Infrastructure	40
3.5 Hybrid Structure.....	42
3.6 Encapsulation of Legacy Applications.....	43
3.7 Combination of Java with CORBA.....	45
3.7.1 Platform Independence.....	45
3.7.2 Java Servlets.....	45
3.7.3 CORBA and Java	46
3.8 Summary	47
Chapter 4 Distributed Gear Design Optimisation	49
4.1 Introduction.....	49
4.2 Development, Integration and Communication of Distributed Components.....	51
4.2.1 Wrapping Structure Based on CORBA	51
4.2.2 Developing Procedure of Distributed Components	52
4.2.2.1 Definition of Object Interface	53
4.2.2.2 Generating Client Stubs and Server Skeletons	54
4.2.2.3 Implementing the Client.....	54
4.2.2.4 Developing the Server and the Object Implementation	55
4.2.2.5 Building.....	55
4.2.3 Integration Model of Distributed Objects	56
4.2.3.1 Different Formats of Java Clients Sharing an Object	56
4.2.3.2 A Client Application Invoke Multiple Objects	58
4.2.4 Management of Objects	59
4.2.4.1 The Naming Service.....	59
4.2.4.2 Portable Object Adapter (POA)	61
4.3 Distributed System Framework of the Gear Design Optimisation	62
4.3.1 Basic Requirements.....	62
4.3.2 Architecture of the Distributed System.....	63
4.3.3 Remote Invocation Mechanism.....	65
4.4 Working Procedure of Distributed Gear Design Optimisation	66

4.4.1 Design Parameters Input	67
4.4.2 Design Evolution by Genetic Algorithm	67
4.4.3 Objective and Constraints Evaluation	68
4.4.4 Design Visualisation	69
4.5 Automatic Gear Tooth Generation.....	70
4.5.1 Gear Profile Calculation Formula	70
4.5.2 DXF Data File Format of the Gear Tooth Profile	72
4.6 Implementation of GUI on Windows.....	72
4.6.1 Design Objectives Identification.....	72
4.6.2 Design Variables Configuration.....	73
4.6.3 Specify Other Design Requirements.....	74
4.6.4 On-line Help.....	75
4.6.5 Input Data Selection.....	75
4.6.6 Calling Optimiser Program	76
4.7 Implementation of Evaluation Programs as CORBA Object on Linux	77
4.8 Implementation of Algorithm Application as CORBA Client on Windows	78
4.9 Summary	80
Chapter 5 Gear Design Optimisation Using Genetic Algorithms.....	82
5.1 Introduction.....	82
5.2 Gear Design Optimisation Model	83
5.2.1 Addendum Modification Design.....	83
5.2.1.1 Addendum Modification	83
5.2.1.2 Effect of Addendum Modification on the Tooth Form and its Application.....	85
5.2.1.3 Consideration of Determining the Addendum Modification Coefficients	86
5.2.2 Gear Design Optimisation Model Considering Addendum Modification ...	87
5.2.2.1 Design variables	88
5.2.2.2 Objectives.....	88
5.2.2.3 Constraints	89
5.3 Implementation of Genetic Algorithms Program.....	91
5.3.1 Basic Concepts of Genetic Algorithms	92

5.3.1.1 General Procedure of Genetic Algorithms	92
5.3.1.2 Operations on Chromosome.....	93
5.3.2 The Cascaded Genetic Algorithm	94
5.3.3 Chromosome Representation	95
5.3.3.1 Chromosome Encoding of Gear Optimisation Application.....	95
5.3.3.2 C++ Bits-field Structure for Chromosome Encoding	97
5.3.4 Variable Dimensional Problems	100
5.4 Multiple Objective Optimisation.....	103
5.4.1 Definition of Multiple Objective Optimisation.....	103
5.4.2 The Weighted Sum Solution	104
5.4.2.1 The Weighted Sum Approach.....	104
5.4.2.2 Fitness Normalisation	105
5.4.2.3 Multiple Objective Optimisation Results of Gear Design	106
5.5 Variable Penalty	107
5.5.1 The Variable Penalty Function.....	108
5.5.2 The Temperature Schedule for Variable Penalty	109
5.5.3 Constraint Calculation.....	109
5.5.4 Instances of Calculation and Results Analysis.....	110
5.5.4.1 Effectiveness of Variable Penalty Approach	110
5.5.4.2 Effect of Temperature Schedule.....	112
5.5.4.3 Effect of Initial Temperature Parameter	112
5.5.4.4 Effect of Fixed Temperature	113
5.5.4.5 Cascaded Structure.....	114
5.6 Summary	116
Chapter 6 Remote Invocation of Single Large Size of Program.....	118
6.1 Introduction	118
6.2 Internet Solution for Executing a Singular Large Program	120
6.2.1 Standalone Design Application Package	120
6.2.2 Internet Solution.....	122
6.3 Development of the System	124
6.3.1 HTML File	124
6.3.2 Servlets.....	126

6.3.2.1 Servlet Vs CGI	126
6.3.2.2 Servlet Running Environment.....	127
6.3.2.3 Implementation of Servlets	128
6.3.3 Applets	130
6.3.3.1 Features of Applets	130
6.3.3.2 Data Retrieval	131
6.3.3.3 The Progress Bar Applet	132
6.3.3.4 Graphics Applet	133
6.3.4 Invoking the Application Object from the Main Servlet	133
6.3.4.1 Defining the IDL Interface.....	134
6.3.4.2 Compiling the Interface into Java ORB and C++ ORB.....	134
6.3.4.3 Developing the CORBA Application Server and the Object Implementation	135
6.3.4.4 Developing the Client	136
6.4 Multi-users Environment	136
6.5 Summary	138
Chapter 7 Applet / CORBA Based Worm Gear Design	140
7.1 Introduction	140
7.2 Architecture of the System.....	141
7.2.1 CORBA IIOP and WWW HTTP	142
7.2.2 Visibroker Gatekeeper	143
7.3 Development of the System	144
7.3.1 IDL Interface Definition	144
7.3.2 Applet Client	146
7.3.3 Object Server.....	146
7.3.4 Object Implementation.....	147
7.4 Deploying and Running the Programs	147
7.5 Summary	149
Chapter 8 Results and Discussions	151
8.1 Introduction.....	151
8.2 Development of the Web-based Architecture	153

8.3 Paradigm 1: Distributed Gear Design Optimisation Using Web Technology and Genetic Algorithms.....	156
8.3.1 The Architecture of the System.....	157
8.3.2 Heterogeneous Application Communication Implementation.....	158
8.3.3 Unified Graphical User Interface (GUI) Design.....	160
8.3.4 Optimiser implementation.....	161
8.3.4.1 Gear Design Optimisation Model	162
8.3.4.2 Cascaded GA Algorithm Design.....	163
8.3.4.3 C++ Bits-field Structure for Chromosome Encoding	163
8.3.4.4 Variable Dimensional Problems	164
8.3.4.5 Variable Penalty Function Using Simulating Annealing	164
8.3.4.6 Multiple Objective Optimisation	165
8.3.5 Automatic 2D Mating Gears Profiles Generation.....	166
8.4 Paradigm 2: Remote Invocation of Singular Large-scale Computing Program Using Servlet and CORBA	166
8.5 Paradigm 3: On-line Worm Design Using Applet/CORBA	167
Chapter 9 Conclusions and Future Work.....	169
9.1 Conclusions	169
9.2 Contributions to Knowledge	174
9.2.1 Conceptual Contributions for Web-based Structure	175
9.2.2 Technical Contributions	176
9.3 Future Work	177
9.3.1 Improving Infrastructure Based on Emerging Web Technologies	177
9.3.2 Multiple Disciplinary Optimisations Based on Distributed System	179
9.3.3 Semantic CAD Environment.....	179
9.3.4 Adaptation of Graphical Data	180
References	181
Appendix A The Main Features of CORBA.....	191
A.1 The Object Request Broker (ORB).....	191
A.2 The Interface Definition Language (IDL).....	192
A.3 Static Stubs and Skeletons.....	192
A.4 Dynamic Invocation and Dispatch	193

A.5 Object Adapters.....	193
A.6 Inter-ORB Protocols.....	194
Appendix B Java Platform Independence	195
Appendix C Involute Spur and Helical Gear Design	197
C.1 Basic Knowledge of Gears	198
C.1.1 Gear Type	198
C.1.2 Properties of Involute	199
C.1.3 Basic Geometrical Parameters of Gears	200
C.1.4 Basic Rack Profile	205
C.2 Addendum Modification	206
C.2.1 Addendum Modification	206
C.2.2 Application Types of Addendum Modification	206
C.2.3 Necessary Calculations for Addendum Modification	209
C.2.3.1 Addendum coefficient h_a^*	209
C.2.3.2 Centre Distance	210
C.2.3.3 Assessment of Bending Stress.....	211
C.2.3.4 Contact stress.....	212
C.2.3.5 Specific Coefficient at Both the Pinion and Wheel Gear	212
C.2.3.6 Constraint Conditions.....	214
C.3 Summary.....	217
Appendix D Basic Methods Controlling the GA Process.....	218
D.1 Chromosome Representation	218
D.2 Selection, Crossover and Mutation	218
D.3 Population.....	222
D.4 Termination of the Optimisation Process.....	222
Appendix E CORBA Model Transition towards Web Services	225
E.1 Introduction.....	225
E.2 Web Service.....	225
E.3 Comparisons between Web Services and CORBA	228
E.3.1 WSDL & IDL	229
E.3.2 IIOP & SOAP	229
E.3.3 CORBA Services & UDDI.....	229

E.3.4 Developing Model	230
E.3.5 Advantages and Disadvantages	230
E.4 Building Web Services from CORBA.....	233
E.5 Intelligent Engineering Design Web Services.....	234
E.6 Developing Issues.....	236
E.6.1 Definition of Each Web Service	236
E.6.2 Intelligent Design Broker	236
E.7 Summary	237

List of Figures

Figure 1.1 Traditional and modern product development.....	2
Figure 1.2 Problem-solving model of collaborative design	3
Figure 2.1 Design process	12
Figure 3.1 Web server centralised architecture.....	34
Figure 3.2 Point-to-point architecture	35
Figure 3.3 CORBA based point-to-point communication	36
Figure 3.4 Common Object Request Broker Architecture.....	38
Figure 3.5 Architecture for the Internet based design environment.....	41
Figure 3.6 The hybrid architecture.....	42
Figure 4.1 Encapsulating structure.....	52
Figure 4.2 Procedure from IDL to the establishment of final client/server structure	53
Figure 4.3 Interface definitions for the existing stress application	54
Figure 4.4 Different clients sharing an object.....	56
Figure 4.5 One common client GUI program invoking multiple objects	58
Figure 4.6 Objects hierarchical naming	59
Figure 4.7 Clients and servers interact with a name server.....	60
Figure 4.8 A POA's role in Client-Object communication.....	62
Figure 4.9 Architecture of distributed collaborative design optimisation.....	63
Figure 4.10 Remote invocations based on the distributed system	65
Figure 4.11 Working procedure of DGDO	67
Figure 4.12 Optimisation procedure and remote evaluation programs invocation	69
Figure 4.13 An example of an automatically generated 2-D gear profile.....	70
Figure 4.14 A 3-D solid model of gears.....	70
Figure 4.15 Gear tooth profile.....	71
Figure 4.16 Design objectives identification	73

Figure 4.17 Selected and unselected optimisation variables.....	74
Figure 4.18 Application, quality and material parameters data input.....	74
Figure 4.19 Pop-up on-line help	75
Figure 4.20 Input data selection.....	75
Figure 4.21 Retrieve Existing Design Parameters file open dialog	76
Figure 4.22 Calling algorithm program	76
Figure 4.23 Registration of remote naming server	79
Figure 4.24 Client procedure is added in a “Process”.....	80
Figure 5.1 Four instances of addendum modification.....	84
Figure 5.2 Schematic diagram of the proposed Cascaded GA approach.....	94
Figure 5.3 Pareto solutions of centre distance and actual contact stress.....	106
Figure 5.4 Pareto solutions of actual contact stress and actual pinion bending stress	107
Figure 5.5 Variation of the calculated centre distance with generation, ten runs	111
Figure 5.6 Variation of the calculated centre distance with generation, one run	111
Figure 5.7 Effect of varying the function for the cooling schedule	112
Figure 5.8 Effect of varying the initial temperature parameter.....	113
Figure 5.9 Effect of using a fixed temperature on the convergence process	114
Figure 5.10 Convergence process for two tiers cascaded structure	115
Figure 6.1 Structure of the original gear optimisation software	121
Figure 6.2 Hybrid architecture of the system.....	123
Figure 6.3 Parameter input form	125
Figure 6.4 Progress bar of the remote program execution.....	129
Figure 6.5 Text results shown by the result servlet.....	130
Figure 6.6 Graphical results shown by the result servlet.....	130
Figure 6.7 NTU_DesignCentre.idl.....	134
Figure 6.8 Data file conflict between different users.....	137
Figure 6.9 Solution for data file conflicts in multi-user situation.....	137
Figure 7.1 Overview of the proposed system	142
Figure 7.2 CORBA application extends across the firewall with maintaining the integrity and security of the network.....	144

Figure 7.3 The compiling sketch map of client and server interface definition WormDesign.idl	145
Figure 7.4 The C++ files generated by the idl-to-C++ compiler.	147
Figure 7.5 The GUI of the worm gear design application from the Web browser	149
Figure B.1 Traditional compiled programs	196
Figure B.2 Java programs	196
Figure C.1 Spur and helical gears	199
Figure C.2 Involute curve of gear profile.....	200
Figure C.3 Spur gear geometry	201
Figure C.4 Geometrical parameters of helical gear	204
Figure C.5 Relation between spur gear, basic rack and rack shaped tool.....	205
Figure C.6 Application types of addendum modification.....	207
Figure C.7 Mating gears with centre distance modification y.m	208
Figure C.8 Slide specific sliding ratio.....	213
Figure C.9 Undercutting occurring	214
Figure D.1 Parent selection roulette wheel	219
Figure E.1 Component architecture in a Web Service.....	227
Figure E.2 Generation of client and server components from interface for Web Services and CORBA	230
Figure E.3 Exposing deployed CORBA ORB components as Web Services	233
Figure E.4 Framework of intelligent engineering design Web services	235

List of Tables

Table 5.1 Encoding and decoding values of gear design variables.....	96
Table 5.2 A chromosome with all 9 gene segments.....	101
Table 5.3 Dynamic mapping array and its values	102
Table 5.4 Result comparison of cascade and non cascade structure.....	115
Table C.1 Gears applications	198
Table C.2 Standard normal modules	202
Table C.3 Face width ratio	204
Table E.1 Client and server components in different RPC architecture	228
Table E.2 Transport protocol components of CORBA and Web Services	229

Chapter 1 Introduction

This chapter gives a thorough introduction to the research undertaken for this project. The background to the research is explained, followed by a detailed statement of collaborative design requirements, the project's aim and objectives, and research approach. A brief summary of the chapters contained within this thesis concludes this chapter. The chapter summaries and highlights the discoveries and achievements attained within the course of the research investigation. These achievements are explained relative to their respective applications.

1.1 Background

Strong competition has motivated a need for a new generation of product design environments, where expectations of improved product quality, lead-time, cost and functionality have risen to a level that may not be sustained by a conventional design to manufacture sequence. To remain internationally competitive, adopting a concurrent methodology is vital and requires a collaborative effort from a broad range of disciplines. Figure 1.1 compares traditional and modern product development. In the modern concurrent and collaborative design model, part of the downstream activities in manufacture and sales are merged into the design stage, in order to obtain high-quality and low-cost product design.

Many advanced computerised tools such as Computer Aided Design (CAD) and Computer Aided Manufacture (CAM) systems have been successfully used for over two decades and still play important roles in product design and manufacture nowadays. The CAD/CAM systems enable engineers to design, analyse, simulate, and test products by means of digital prototyping in the early stage of the product development process without actually manufacturing physical parts. These tools

dramatically help companies to produce higher quality products in a shorter time at lower cost.

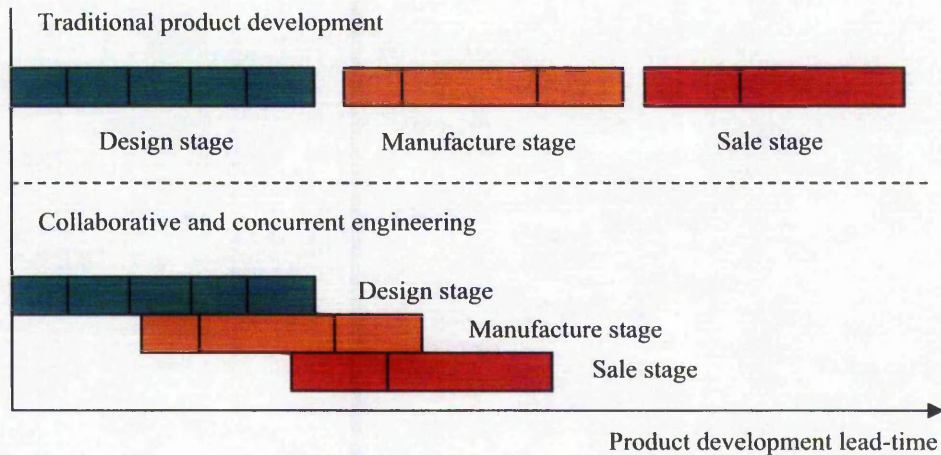


Figure 1.1 Traditional and modern product development

With globalisation of more industrial enterprises and increasing complexity of products, the experts from different domains may belong to different organisations, and even geographically located in different areas over the world. Thus it is necessary to create a distributed and collaborative design environment over the Internet to make the design resources (CAD/CAM systems, other design applications, database files, etc) effectively shared, quickly accessed, and easily and flexibly used when the designers need them. Constructing such a design system will need to overcome the following difficulties:

- **Heterogeneity:** dispersed design applications may use different programming languages, such as C/C++, Python, Java, Perl, or Cobol, and run in different computing platforms such as UNIX, Microsoft Windows, IBM OS/2, or Apple Macintosh.
- **Distribution of both computers and users:** computer network technology has led to a distributed working environment where an ever-increasing number of distributed application systems are involved and running on many computers. These computers are connected over networks having different architectures, protocol standards, bandwidths, etc.

Such a collaborative environment enables multiple specific domain experts and their computerised software tools to work together to search one design space, in order to obtain a common solution. Figure 1.2 shows the problem-solving model of collaborative design in this circumstance. In the present research, the design and optimisation of gears is taken as an example of a distributed application.

Collaborative design aims to find an optimal design solution to a design problem. However most design problems are difficult and complex and affections of the models from different specialists to the design solution are comprehensive. The compromise between the various objectives is not easy to find since the optimisation criteria are often contradictory. Therefore, in the collaborative design environment, not only an appropriate architecture where different experts and their applications can be integrated, but also a resolution strategy to balance these different interests from different domains in such a complex situation is required to ensure efficient and effective collaborative design.

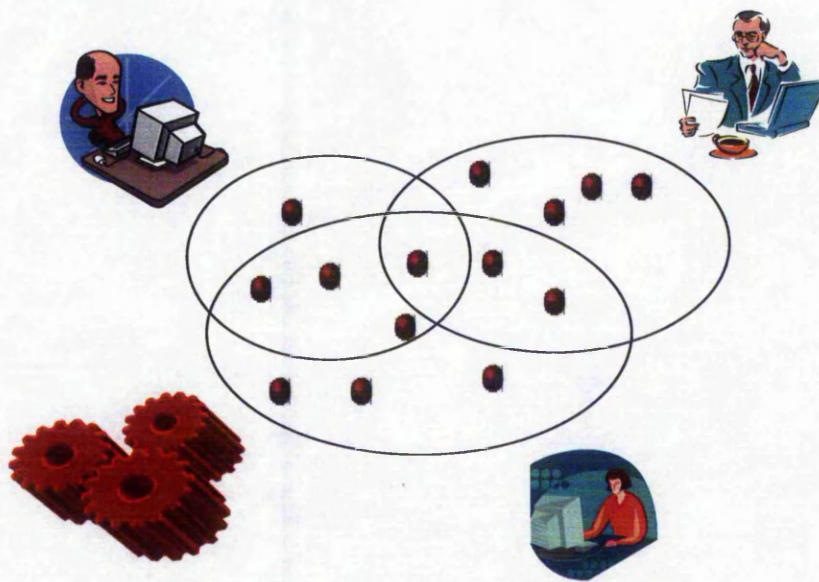


Figure 1.2 Problem-solving model of collaborative design

1.2 Collaborative Design Requirements

To implement this research work, the requirements of collaborative design and the limitations of current systems must first be identified. Engineering collaborative design

is a very broad term that is closely related to several technical and social disciplines such as engineering design, computer science, human computer interaction, decision-making behaviour, and social sciences. In this research, the term “collaborative design” refers to a design activity where distributed applications will be shared and invoked by an active main designer towards a complex design problem over the Internet. With regard to the application heterogeneity and distribution, the following features of building a collaborative design system will be the challenges in this research:

- **Legacy system reusability:** system development cost is an inevitable problem to be taken into consideration. Reuse of legacy systems is a methodology for reducing the high cost of software development and maintenance. It is necessary to build an environment to accommodate many existing design and manufacturing applications without rewriting the essential codes.
- **Scalability:** scalability means that additional resources can be incorporated into the system as required. This capability should be possible without disrupting the links previously established.
- **Resource sharing and management:** The system design should support the finding, accessing, and management of design sources.

From the view of engineering design, some problems need to be resolved, such as:

- **Remote execution of a single large-size computing program:** in the engineering field, there are many large-size computing programs for product design and analysis. These programs may not be convenient or not allowed to be downloaded on the client machine and likely run on the owner’s previous machine. System design needs to facilitate the parameters input, program execution and monitoring, and result viewing.
- **Invocation of multiple programs:** to a complex product design, there may be a need for invocation of multiple programs during a complex design procedure. These applications should be available over the Internet and can be invoked dynamically according to the user requirement.

- Interest leverage between multiple design domains: to a single design problem, multiple interests from different design domains should be leveraged especially when they are controversial.
- Quick reaction to alternative design modification: system design should provide an integrated environment to implement the parameterised design mechanism. To each modification of design input, system can be facilitated with viewing of resultant data, geometrical shape creation, manufacturing check and motion simulation.

Most of the existing CAD/CAM systems are standalone, completely autonomous, and single-user applications. Some CAD/CAM packages, such as Windchill for Pro-Engineer, enable the users to collaborate over the Internet. However, the Internet-based collaboration can only be operated with the same software in the same type of platform and operating system. Therefore, there is inevitably for a need to construct a distributed and collaborative environment to accommodate heterogeneous and standalone resources to conduct a design.

The observed limitations of current CAD/CAM systems and the requirements of engineering collaborative design have led to the following formulated research question in this thesis:

Which tools, methods, and architectures are needed to develop the distributed system supporting engineering collaborative design over the Internet?

This is a broad research question that can be further divided into a few major sub-questions, namely:

- How should huge amounts of complex engineering design resources, which are distributed, heterogeneous, dynamic, and continuously evolving, be efficiently integrated?
- How do engineering design resources communicating with each other?
- How is collaboration between multiple distributed applications conducted to resolve a design problem?

The combination of all the above features to provide a robust tool for collaborative environment applied in design and manufacture is a novel and challenging task. One task in this project is to investigate and choose proper Internet and Web technologies to develop such a collaborative environment for integrated design at low cost, with the features of heterogeneity, scalability, interoperability and legacy component reusability. Another task is to provide a proper optimisation mechanism to support collaborative design conducted by multiple designers. As a case study, spur and helical gear design based on the proposed system is implemented.

1.3 Aims, Objectives and Scope of the Research

The overall aim of this project is to construct a Web-based system at low cost, to support effective collaborative design over the Internet and thereby shorten the product development lead-time. More specifically, the following main objectives in support of this aim are:

- To review current relevant work and investigate enabling Web technologies and tools.
- To establish an infrastructure of Web-based collaborative environment for integrated design.
- To develop an appropriate optimisation mechanism for the co-design among multiple concerns.
- To develop an approach to invoke singular large sized design programs over the Internet and multiple users management.
- To develop an architecture to support collaborative work of multiple heterogeneous applications.
- Taking gear design as a case study, to explore the feasibility of the proposed system.

This thesis is grounded in the field of mechanical engineering and draws attention to some technical issues of constructing distributed system for engineering collaborative design. In this research, existing Web-based technologies such as Java and distributed object computing are utilised to achieve the research aim. The optimisation approaches are employed to assist a designer to conduct the design optimisation over the

distributed collaborative design concerns. It is expected that the findings can provide an efficient mechanism for engineering collaborative design over the Internet.

1.4 Research Approach

This research work employs distributed object computing technology to build a collaborative environment supporting integrated design, an optimisation method to implement collaborative co-decision among different domain concerns, Web-oriented programming tools such as Java to develop friendly graphical user interface within web browser, and C++ programming tool to develop the application interface for existing C++ design applications.

In order to create, deploy, and manage the distributed components, CORBA (the Common Object Request Broker Architecture), a well-known specification for distributed object computing, is used to implement design resource integration in a platform, hardware, and software-independent manner. Existing design applications can be wrapped into distributed objects, irrespective of which language it is written, which platform it is running on and what hardware it is ported on. Proper CORBA developing tools need to be investigated in order to fulfil those communications between C++ applications and Java applications, and between applications residing on Windows and on Linux.

In addition, Genetic Algorithms (GA) are utilised in this research. With respect to the concurrent and collaborative engineering, there may be many designers from a wide range of disciplines to carry out one single design problem. Affections to the design parameters from different domain consideration are complex, and sometimes even controversial. GA-based design optimisation procedure is modelled to help to evaluate the multiple design objectives and to find multiple feasible inferior solutions with global searching.

Furthermore, Java Applets and Java Servlets technologies are used to fulfil the remote execution of large sized and time-consuming computing program. Servlets, as a Web server extension, are used for accepting user client, activating execution, and returning

results with a Web page. Java Applets can be embedded in a Web page and used to design a friendly user interface. Applets are also combined with CORBA to provide a more powerful model for a distributed system.

Moreover, the C++ programming language is used to develop some applications such as a graphical user interface program on Windows and an algorithm execution program on Linux. In the proposed distributed system, there are many existing applications written in C++ to be integrated. The service program for these programs needs to be developed in C++.

1.5 Outline of This Thesis

Chapter 2 Review for Web-based Collaborative Design

Research and literature relating to the major topics of the project are investigated and reviewed. The topics to be reviewed are Internet-based communication methods and the improvement in the field of engineering design. Traditional design methods with respect to sequential and much iterative process for product development and emerging concurrent and collaborative engineering to improve the product developing process are reviewed. Current collaborative systems and relevant technologies are investigated. Also, data exchange technologies and the Genetic Algorithm technique are explored to identify their capabilities with respect to product information exchange and design optimisation in the field of engineering design.

Chapter 3 Overview on Architectures of Distributed System

In order to provide a Web-based collaborative environment for integrated design, two architectures of distributed system: point-to-point structure and server-centralised structure are investigated and described in this chapter. The application models of these two architectures are presented. The combined model of them is also proposed in order to provide an effective and easy-to-administrate platform for the distributed application system. Architecture and technologies to implement the distributed system are explored. A combination of CORBA and Java is considered to address the challenges.

Chapter 4 Distributed Gear Design Optimisation

Gear design is a complex process and often needs collaborative design conducted by different disciplinary experts. This chapter presents a distributed environment based on CORBA, all the computerised program or service procedure owned by the experts are wrapped into distributed components and consequently can be integrated together, and communicate with each other. All the concerns from different design experts are abstracted as objectives or constraints in the design optimisation model. An optimising mechanism, i.e. optimiser, supporting design optimising and design result visualisation is developed and illustrated.

Chapter 5 Gear Design Optimisation Using Genetic Algorithms

In order to implement the design optimisation, an appropriate optimisation approach is needed. In this research, Genetic Algorithms (GA) is utilised. In this chapter, the GA approach is investigated, followed by the modelling for the addendum modification design problem of spur and helical gears, and then key issues about controlling the optimising process are presented. Variable penalty function for controlling the convergence process, dynamic variable combination for the variable dimension problem, and multiple objective optimisation approach for the co-design model are illustrated.

Chapter 6 Remote Invocation of Single Large Size of Program

Integration and remote invocation of a large size of program is developed and presented in this chapter. The implementation of parameter input, program execution and monitoring, and result visualisation within a web page is demonstrated. Multiple user environments are developed and described as well. The system is developed based on a server-centralised model using Java Servlet and CORBA. In this model, the execution of a program is not affected due to network interruption and calculation results can be viewed later.

Chapter 7 Applet / CORBA Based Worm Gear Design

By using CORBA and Java applet, a plug-in running model of a remote program for worm design is given. A session established between point-to-point can be kept linking

until stopped manually. This model is used for the flexible need for parameter modification repetitively.

Chapter 8 Results and Discussions

This chapter discusses the features of the Web-based systems, the framework and developing issues about three application paradigms: the distributed gear design optimisation with GA-based optimiser, remote invocation of singular large-scale program, and the worm gearing design system. Transformation of a developed system towards the emerging model is also summarised. Along with the discussion of work, the achievements are accordingly presented in every part of work.

Chapter 9 Conclusions and Future Work

This chapter gives the conclusions from the research work with the descriptions of contribution to knowledge. Further research work will be further conducted in several aspects: explorations on Web technologies, extension towards multiple disciplinary optimisation, semantic CAD environment and adaptive graphical data for mobile environment.

Chapter 2 Review for Web-based Collaborative Design

This chapter highlights the benefits and drawbacks of current collaborative systems, current Web-technologies that are relevant to collaborative design are presented, and the similarity between collaborative architectures and Web-technologies. A number of web-based collaborative design environment are reviewed. Finally, current challenges and hot topics are discussed.

2.1 Fundamentals of Collaborative Design

2.1.1 Design Process, Concurrent Engineering & Collaborative Engineering

Product design is the process by which the needs of the customer or the marketplace are transformed into a product satisfying these needs. Five main specific dimensions, all of which ultimately relate to profit, are commonly used to assess the performance of a product development effort: product quality, product cost, development time, development cost, and development capability [1].

Typically, the design of a new product consists of six basic design stages, illustrated in Figure 2.1.

All design starts with the identification of a need. Ideas for new products or product improvements usually come from some combination of focus groups, customer feedback, market surveys, published studies, and company intuition.

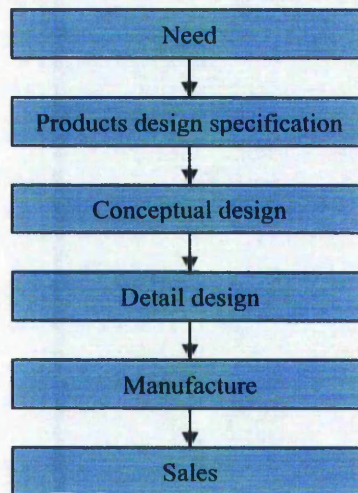


Figure 2.1 Design process

Once a market need is identified, it is needed to formulate product design specifications (PDS) such as performance parameters, required features, appearance, size, weight, user, cost, environments for storage and use, maintenance requirements, expected life, shipping methods and so on.

Designers use the results of specification definition phase to generate and evaluate concepts for the product. Generating concepts involves establishment of the functions to be included in the design, identification and development of suitable solutions. The goals to evaluate concepts are to compare the concepts generated to the requirements developed during specification development and to select the best concepts for refinement into products.

Once the engineers are satisfied that the concept is technically feasible the detail design can proceed in earnest. During this stage, each part is identified and engineered. This detail work will involve in-depth decisions as to the geometry, tolerances, materials, surface finish, etc. It will also include decisions as to the method of manufacture, the materials to be used, the need for special purpose machinery of tooling, whether existing components can be adapted, the advisability of making in-house or buying in, and so on. The design is documented with drawings and computer

files describing the geometry of each part and the process plans for the fabrication and assembly of the product.

Manufacture is used as a synonym for production, that is, to refer only to the portion of the product realization process that involves the actual physical processing of materials and the assembly of parts.

Sales include the marketing of the final product – distribution, service back-up, etc., which are all part of the marketing or selling activity.

Conventionally, the above-mentioned activities are divided into almost separate functions, resulting in design and manufacture occurring as serial activities. Product designers are mainly concerned about their products' performance and functionality and may not give sufficient consideration of process design, manufacturing's constraints and/or marketing's constraints. In recent years, it has been found that sequential design is brittle and inflexible and often requires numerous iterations, which make the design expensive and time-consuming, and also limit the number of design alternatives that can be tried out. Sequential design is usually practiced with downstream information flow. It may cause an inefficient design (and hence inefficient product development), due to the absence of manufacture-ability checks at the design stage.

Due to the growing demands for better and cheaper products, the traditional practice of passing decisions sequentially "over the wall" has been replaced with the concurrent engineering approach in recent years. In this approach, the simultaneous design of a product and all its related processes in a manufacturing system are taken into consideration, ensuring required matching of the product's structural requirements with functional requirements and the associated manufacturing implications. This means that all pertinent information flows should be multi-directional among the design functions and all related processes in an organization. Processes influencing product design usually include market analysis, materials procurement, product cost estimation, machining, assembly, inspection as well as the later phases of the product's life cycle such as service, maintenance, disposal and recycling [2].

Many aliases of concurrent engineering have been used to describe similar approaches, including simultaneous engineering, life-cycle engineering, design integrated manufacturing, design for X (DFX), parallel engineering, concurrent design, design fusion, and design in the large. Design for X stands for all the other related upstream and downstream functions in the development process of product, such as Design for Manufacture (DFM), design for producibility, design for assemblability, design for serviceability, and so on [2].

With the increasing availability of world wide concurrent grid of engineering resources, the concurrent engineering approach has delivered significant impact on industrial productivity over the past decade. However, as the product complexity increases and engineers are asked to be more socially responsible, engineering decisions, whether they are made sequentially or concurrently, must involve multiple stakeholders with different expertise and competing interests. Facing with this additional challenge engineers must have new methods of not only sequencing multiple decisions but also negotiating a single agreement. The task of achieving a jointly agreed decision among multiple stakeholders is called Collaborative Engineering, which represents the next frontier of concurrent engineering research [3].

According to the definition given by Mills [4], collaborative engineering is the application of team-collaboration practices to the organization's product development endeavours. It builds upon the nature of cross-functional product development teams introduced in the realm of concurrent engineering. While concurrent engineering has historically been concerned with the careful structuring of products, workflow, teams, and organizations, collaborative engineering is, in contrast, more concerned with creating the environment of effective, free-flowing, and ad-hoc collaboration among peers whose insights complement one another and whose whole as a team is far greater than the sum of its individual parts.

When a product is designed through the collaborative engineering and collective efforts of many designers, the design process may be called Collaborative Design (it

may be also called Co-operative Design or Inter-disciplinary Design). The objectives of the collaborative design team might include optimising the mechanical function of the product, minimizing the product developing costs, or ensuring that the product can fast, easily and economically be serviced and maintained [5].

Based on collaborative design engineering, traditional approaches with large, isolated groups of engineers and designers have given way to small, nimble, multidisciplinary integrated product teams that are able to quickly catch and correct design flaws. By increasing communication and coordination, these new teams can more quickly and at reduced costs complete the development of high-quality products [6].

2.1.2 Collaborative Design & the Internet

Primarily, collaborative design tries to address those problems in traditional sequential models for design generation, concurrently by considering constraints and detecting conflicts early in the design stage, as concurrent engineering. It might include those functions as disparate as design, manufacturing, assembly, test, quality and purchasing as well as those from suppliers and customers [7]. Wang [8] introduced a unique combination of machining feature, agent technology, and function block to tackle and facilitate concurrent design problems with due considerations of downstream constraints under distributed environment.

However, collaborative design engineering not only augment the capabilities of the individual specialists in the structural combination, but as an extension of concurrent engineering, must also enhance the ability of collaborators to interact with each other and with computational resources, whether in a sequential or concurrent working flow. Chung developed a framework called Manufacturing Integration and Design Automation System (MIDAS), where engineers could (a) identify valid process configurations based on constraints provided from many different sources; (b) update and share the process configuration via Internet; and (c) facilitate coordination among distributed process participants [9]

To implement product development based on collaborative design engineering, the people standing for multiple-domain experts and their computerised technical tools such as computer-aided design, analysis, or manufacture tools, product information files, knowledge bases, and so on are essential elements for collaboration. A proper infrastructure what makes collaboration possible is also important for gaining required collaboration. In essence, collaborative design engineering is all the things about establishing an environment including these three elements, i.e. people, tools and infrastructure, to support collaboration.

In this context, the Internet becomes a unique infrastructure for collaborative design engineering. Nowadays, it is very difficult to talk about concurrent or collaborative design and manufacture, or virtual enterprises or alliances without mentioning the enabling information infrastructures built over the Internet. Remote collaborative design and manufacturing becomes a specific form of virtual alliance, which exploits the Internet technologies for information communication and function coordination among design partners scattered over long distances. The ever-increasing improvement of Web technologies has opened more and continuing possibilities for resource integration and access, data sharing, and collaborative design generation required in collaborative design engineering.

Significant efforts have been made to develop computer supports and information technologies to facilitate collaborative teamwork. Early developments and achievements in computer supported concurrent engineering had been reported in an American Society of Mechanical Engineers (ASME) workshop organised by Sriram, Logcher and Fukuda [10] and a special issue in the IEEE (Institution of Electrical and Electronic Engineers) Computer Journal (1993). Recent Surveys by Huang and Mak [11] and Erkes et al. [12] indicate that the Internet/Web technology is playing increasing roles in developing and applying Collaborative product design. Indeed, a number of major initiatives and projects recently launched in America and Europe involving government, academic and industrial institutions, have adopted the Internet/Web technology as their development infrastructure. Examples include the Manufacturing Automation and Design Engineering (MADE) program and its follow-

up Rapidprogram [13], the Agile Infrastructure Manufacturing Systems (AIMS) project [14], Technologies Enabling Agile Manufacturing (TEAM) program [15], the Global Engineering Networking (GEN) project [16], Production Planning and Management in an Extended Enterprise (PRODNET) project [17] and the Web-enabled Collaboration in Intelligent Design and Manufacture (WCIDM) project [18]. More projects are emerging in other countries and regions. Demonstrative systems are being developed. CyberCut [19] and MADEFAST [20] have also presented experimental workbenches for Web-based design to production, including activities such as conceptual and detail product design, process planning, Design for Manufacture, NC (Numerical Control) programming and rapid prototyping.

In parallel, collaborative design is also functionally supported and improved by the technologies in the domain of artificial intelligence, such as agent technology, knowledge management, knowledge-based systems, genetic algorithm, and so on. Furthermore, product data format and exchange technologies have been accordingly evolved and developed for Web-based collaborative engineering. These enabling technologies research serve as the wheels of the collaborative design vehicle to more forward [21].

2.2 Web-based Collaborative Design

The Internet is a large and connected network of computers and the World-Wide Web (WWW) is the fastest growing segment of the Internet [22]. The popularity of the Internet is largely due to the influence of the World Wide Web proposed in 1989, which has made the Internet accessible and available to mass population. Powered by the ever-improving information technologies, such as HTML (HyperText Markup Language), Java, search engines, email, XML (eXtensible Markup Language), and distributed object technology, the Web provides an interface to feel information for interaction. The ability of the Web for designers to combine multimedia to publish information relevant to the spectrum of the design process, from concept generation and prototyping to product realisation and virtual manufacturing, motivated the adoption of the Web as a design collaboration tool. A collaborative design system developed with the Web as a backbone would primarily provide: (1) access to

catalogue and design information on components and sub-assemblies; (2) communication among multidisciplinary design team members in multimedia formats; and (3) authenticated access to design tools, services and documents [21].

The Web technology has been applied or experimented to develop virtually many aspects of design across the entire life cycle of the product development and realisation process. Starting from the front-end, the Web-based approach is particularly suitable for customer requirement management in the new or market research. An interesting work at the Philips Advanced Development Centre [23] is to use the Internet as a communication infrastructure for lead user involvement in the new product development process. Another related project is to use the Web for customer requirement analysis for software product development [24].

The utilization of a web interface allows individuals to access relevant design information [25]. The Web can be used for the design team members as a medium to share data, information and knowledge [26, 27]. In some cases the Web is also integrated with appropriate technologies for product data management. Research papers [28-30] show that 40-50% parts of a new product are entirely the same as existing ones, 30-40% parts require slight modification of existing ones, and only 10-22% parts are new. Thus Wang et al [31] focus on the management of existing design resource including the design knowledge, methods and geometry data of those existing product parts and tried to enable the maximum reuse of existing design resources in the development of a new product.

Many CAD/CAM systems are usually designed to work in isolated environment and not able to communicate with each other. In the research of [32], the framework was built on the Internet for data exchange among different CAD/CAM systems. Translator applications located on the Web is easily used to translate files of different formats into the standard STEP format for other users to share them.

Some contribution is focused on the design process management. Yoo and Kim' Web-based knowledge management system is for facilitating seamless sharing of product

data among application systems in virtual enterprises [33]. MIDAS [34] is a framework for integrated design and manufacturing process. One of facilities of the system is to support a collaborative design by sharing data and processes. Engineers are able to access a process, compare alternative process, analyse interdependencies, and generate their own processes [34]. In the research of [35], the web-based and adaptive design process management approach is software tool based on dynamically constructed 'flows'. A key aspect of the design process management is that multi-perspective visualisation allows both managers and designers to view the design process from their respective perspectives.

In manufacture, Wanger et al [36] experimented with developing fixture design systems on the Internet. Rapid prototyping is one of areas where the Web technology has been applied [37]. Krause et al. [38] employed the Web technology to achieve global product data management. Wang et al [39] presented implementation of remote robot manufacturing over Internet. The application system is under hybrid architecture of Web browser/server and client/server to manipulate the product data and to carry out a variety of manufacturing functions. A well-developed product model should have the following characteristics: encapsulating product data into data objects; capturing all product data required and generated throughout the whole product lifecycles; organizing various product data in a logical way to allow multiple users operate on the same data model, and dynamically reflecting product configuration transformation process through different manufacturing stages.

Different web technologies are used for the different function and at different levels. In the research of developing Web-based concurrent design systems, Name and Engelstein listed tools for distributed concurrent design [40]. These tools include e-mail, World Wide Web (WWW), Virtual Reality Modelling Language (VRML), File Transfer Protocol (FTP), groupware, and so on.

Roy et al. [41] reported the development of a prototype Web-based collaborative product modelling system, in which designers can collaborate through shared Web pages and VRML models. Li et al [42] proposed the collaborative product

development mode on Internet to allow partners easily to retrieve design information stored in HTML, VRML. Asynchronous collaboration can be received through e-mail and Web publishing among collaborators to get to know design task on demand, problem discussion, newest advanced technical progress, and so on [43].

The Web with only HTML page is not enough for the interactive medium between client and server side. Web servers needs to route the content of HTML forms to back-end server applications. Some technologies such as CGI (The Common Gateway Interface), Java, JavaScript, ASP (Active Server Pages), and so on are emerged into Web application to enhance the interaction ability.

Wu's Cdesign system [43] is built on client /server architecture. Through centralised Web server and its services implemented in Java, ASP applications dispersed knowledge bases, design applications, collaborative management applications, and clients possess Web-browser are linked together to conduct collaborative design.

Chun [44] proposes product recommender system implemented by JESS (Java Expert System Shell) and Java Servlet. The system employs a multimedia user interface that is accessible with standard web browsers. In the client/server system, the client software supports the user interface and the geographically separated server software supports the expert system. The database is accessed using the JDBC (Java Database Connection) capability of the Java Server Extension (Servlet). Product domain knowledge is stored as a form of facts and rules in a JESS knowledge base file.

Ballaminut et al's WIRED is a framework, written in Java, to build High Energy Physics event displays used across the network. It can be used as a stand-alone application or as an applet inside a WWW browser [45].

Abdel-Wahab et al developed a system called Java Collaborative Environment (JCE) in Java application and showed how a group of Internet users can share single-user Java applications for synchronous collaboration. The system allows application sharing

among diverse systems such as UNIX workstation-based and PC windows-based systems [45, 46].

The main problems of these approaches in the above systems are that they require HTTP and the Web server to mediate between objects running on the client and objects running on the server. There is no way for a client to directly invoke a server object. However collaborative application among diverse partners over Internet needs a highly interactive communication between resource components. To get the full benefits of object-to-object interactions, the Web must be augmented with a distributed object infrastructure. This kind of technologies includes RMI (Remote Method Invocation), DCOM (Distributed Component Object Model) or CORBA (the Common Object Request Broker Architecture). RMI is for communication for Java applications; DCOM is for systems based on Microsoft's Windows; and CORBA is designed for language-independent and platform-independent application.

Raje et al's collaborative environment for visualization using Java RMI is developed for the purpose of native Java-client and Java-server. It allows users to view or explore 3D objects, natural phenomenon, or complex data [46].

In the Alda et al's research for supporting collaborative design among multiple universities, DCOM is used for the deployment and invocation of application components over the peer-to-peer architecture [47].

To the complex and highly diverse collaborative design environment, CORBA is more used than others. CORBA, defined by the Object Management Group (OMG), is a standard for the distributed computing and systems integration [48]. Its promises of platform-independent and language-independent enable object components to be operated from anywhere in a network without concern for the operating system they are running on or the programming language they are written in.

In the research conducted by Kim et al, the system integrates multiple clients, application servers, and databases together over a three-tier structure. The

communication between clients and application servers is done via CORBA for interoperability among the distributed objects [49].

Li et al [50] discusses the idea that the collaborative product development mode on Internet is a process of collaborative decision making in stages. CORBA is used to establish an environment for collaborative problem solving.

Ong [51] implemented a multi-tier Web-based environment for accessing power system data via networks, in which CORBA is used for the interface between the Java GUI and Power Flow software. CORBA helps to turn a local PF application into the Web-based simulator tool. This demonstrates how any existing applications written using C/C++ can blend easily with Java. Similar examples are also given in [52-54].

Yoo [55] developed a Web-based knowledge management system for sharing data in virtual enterprises, where CORBA interface helps Java agents communicated with the knowledge base system and in replace of CGI/HTTP techniques. This enables more dynamic user interfaces on heterogeneous Java virtual machines, updating for any changes in the user interfaces.

Hauch et al [56] explored communication between integrated software tools using CORBA. The proposed system allows the encapsulated components in different processes on different machines to directly communicate in a high-level manner. Java is used for the implementation of system components and GUI. A CORBA-compliant wrapping tool is developed for the encapsulation of system components.

Lee's Collaborative optimisation approach for multidisciplinary design optimisations of mechanical systems allows diverse optimising system belonging to different disciplinary co-optimize a single problem. The system level service program is functioned as the client program that requests the necessary information. Each discipline optimiser can be functioned as the server programs, located in the heterogeneous systems connected with network. CORBA offers common interfaces [57].

Sang focused on the CORBA wrapping of legacy scientific applications, especially the procedures in details for wrapping to the non-object-oriented Fortran codes using CORBA and C++ is demonstrated [58].

In the fast-developing IT world, new object and exchange paradigms emerge rapidly. A wide acceptance of approaches based on the extensible Markup Language (XML) appears into the world. The W3C defines a set of XML-based languages that are the foundation for the current notion of web services. The Web Service Description Language (WSDL) issued to describe interfaces of a web service. These interfaces can be accessed using the Simple Object Access Protocol (SOAP), in which recent development contends with CORBA [59].

Ouyang et al [60] presented a novel web service based distributed collaborative CAD system, employing feature as collaborative elements. SOAP is used for the communication between the server and the client. The system supports collaborative design on heterogeneous environment.

Lostienko et al [61] introduced an Advanced Collaborative Infrastructure (ACI) for distance spanning, tool integration, and administration as well as open interfaces for XML-based data exchange. The ACI core components are implemented as Web Services that are interconnected using the SOAP. The SOAP messages between the components are transported using ANTS (Advanced .NET Testing System).

Most of the Web-based collaborative design systems would primarily provide: (1) access to catalogue and design information on components and sub-systems; (2) communication among multidisciplinary design team members in multimedia formats; and (3) authenticated access to design tools, services and documents; (4) design team members would use the Web as a medium to share data, information and knowledge.

However, to implement collaborative design over the Web-based infrastructure, some other enabling technologies, such as data exchange, artificial intelligence, and so on, have to be involved.

2.3 Data Exchange

The CAD/CAM systems have been developed for many years. Most of them are capable of handling complex geometric information for a product throughout its life cycle. However, these systems may not be able to communicate with each other in collaborative design environment. To build up a collaborative environment for CAD/CAM system users on different sites communication requires a proper standard to represent all information to be transferred [62-65] and a good communication system tool to breakthrough the barrier of product data exchange and sharing [66-69].

In the CAD/CAM context, there are several existing standards for data exchange, such as Initial Graphics Exchange Specification (IGES), SET, VDA-FS, EDIF, DXF, etc [70]. The most popular exchange standard in use is the IGES. It was designed as a neutral format for the exchange of CAD data and has been used as the standard for geometric information by most CAD/CAM systems.

Although IGES is best supported as an interchange format for geometric information, it cannot fulfil the completeness requirement in representing product data. STEP was first proposed in 1984 to represent complete information of a product throughout its life cycle. It integrates geometric representation and adds additional information, such as the process models, for different stages of the product development. The information in STEP is represented using an information modelling language called EXPRESS. STEP uses application protocols (APs) to specify the representation of product information for one or more applications [71].

STEP has gained considerable importance due to active support from the automobile, aerospace and the defence sectors. CAD software vendors and third parties for systems like AutoCAD, PRO/Engineer, CATIA, Unigraphics, Microstation, Trispectives and ACIS, have STEP interfaces in their new releases. As the acceptance and the use of STEP-based information exchange increases, a need for translation services to convert information represented in legacy file formats like IGES into STEP will arise. Bhandarkar et al 's report focuses on a procedure for converting product design data

from legacy IGES format into STEP [72]. Chao and Wang developed an application for exchanging CAD/CAM data on Internet. In this research the AutoCAD DXF file format is changed into STEP format [73]. Oh et al presented an interface module based on the STEP methods to implement the data exchange between a CAD system and a PDM system [74].

To implement the data exchange, individual STEP translators for various systems need to be developed. A possible solution to this problem is to provide the Internet-based services for STEP data translation. Zhang et al demonstrates an avenue for on-line STEP data translation services for virtual enterprises [75].

Hardwick et al. [76] proposed an infrastructure for sharing manufacturing information for the virtual enterprise. They use the STEP model data as the standard to represent product data, the CORBA as the communication tool, and WWW as their infrastructure.

Using STEP, Regli [77] discusses the feature of Internet-enabled CAD systems and brings out two features that Internet tools should have: access to information, access to tools and collaborators. Smith and Muller [8] focused on obtaining a multi-view database system for information sharing for establishing a concurrent engineering environment using STEP. Ly [78] builds a distributed editing system on the network so that the editing processes can be carried out on it.

Brown and Versprille [79] discuss issues on information sharing through the network among different CAx (a summary term for various kinds of Computer Aided technologies) systems. They focus on feature extraction methods of traditional CAD/CAM databases. They use CORBA as the tool to transmit features, and stored them in an object-oriented database system and suggested that other tools such as MicroSoft's ActiveX can be used for transmission. Kimuro et al. [80] discuss a Continuous Acquisition of logistic Support (CALs) environment with CAD databases using agents. They use such technology to query distributed CAD databases as a centralised database system.

Chao and Wang focused on establish a data exchange and sharing environment for distributed CAD/CAM users across different platforms and implemented a framework including client databases, an index server, a CAD data format translator, and a file sharing control module to provide engineers a handy tool to implement concurrent engineering. The CAD data format translator can be used for translate different formats into the STEP format [81].

Chin et al proposed a methodology that represents multiview integrated product modelling based on the outcome of a research project on ISO 10303 STEP. They have proved that the STEP EXPRESS language is applicable for describing the integrated product model, while the mapping has been successfully developed by EXPRESS-X language [82].

Deshayes et al proposed an approach allowing different experts to cooperate via Internet. To facilitate the communication of information among different experts, STEP standards are viable tool that can be used in this frame [83].

Yeh and You implemented a pilot system for STEP-based product Data Exchange system for the requirement of product data exchange between enterprises. The data from miscellaneous information systems, such as CAD/CAM, MRP/MRPII, ERP, and PDM, in the design and manufacturing phase of a product's life cycle can be exchanged using this system [84].

The STEP has been the dominant technology for product data distribution and sharing. It provides a systematic approach for well-established user communities to share data on specific types of products and manufacture. However, when the first STEP standards were developed, the World-Wide Web was in its infancy. Since then, the development of Web technology and its popularity have opened up further opportunities in the application of STEP. Many efforts have gone into making STEP technology compatible with the Internet [75-81, 83]. Especially, in the recent time, the XML is becoming an obvious choice. Integrating STEP with XML facilitates

the deployment of STEP data in the Internet/WWW domain. With the increasing popularity of the XML on the Internet, mapping STEP data into XML becomes a logical way to make STEP data more accessible through the Internet. Chan et al [85] discuss the ways of automated conversion from STEP into XML, and the exchange of STEP data through the XML-based mediator architecture.

2.4 Genetic Algorithms

Genetic Algorithms are stochastic search techniques based on the mechanism of natural selection and natural genetics [103]. GA is an ideal solution to the gear design optimisation. The nature of GA meets the engineering requirements in two aspects: GA does not have much mathematical requirement about the optimisation problems and can handle any kind of objective functions and any kind of constraints (i.e. linear or nonlinear) defined on discrete, continuous, or mixed search spaces. It would accommodate any complex consideration from multiple domains. On the other hand, the ergodicity of evolution operators makes genetic algorithms very effective at performing global search (in probability). It would benefit from a broad region of search and retrieve a wide class of alternative design solution [100].

Genetic Algorithms is used for performance-based design evolution and automatic design of fuzzy system. The approaches allow the user to explore and visualise the design evolution and its form generation in an attempt to stimulate the designer creativity that might contribute to their output [90, 91].

2.5 Other Web-based Technologies

Significant research has focused on technologies that can assist designers for collaborative product development in the distributed design environment. In addition to the Web for the infrastructure and STEP data exchange, those include CAD conferencing, work process modelling and management, agent-based knowledge sharing and conflict management [86].

CAD conferencing supports synchronous collaborative works by exchanging geometric models using a teleconferencing system in a networked environment. Most

of the research in this field focuses on application sharing, co-authoring, three-dimensional geometry visualizing and desktop conferencing [87].

Research for agent-based knowledge sharing has focused on enabling collaboration among software agents [88]. In the SHARE project, knowledge query manipulation Language (KQML) was used for supporting design teams in sharing their understanding of the design process [26].

Conflict management is needed to coordinate information for collaborative design. Coordination strategies that avoid conflicts between design participants have been proposed at the task level [89].

All of the technologies have been implemented in application forms existed in the Web. In the collaborative domain, environment design should include finding and controlling the execution of these distributed design method or rule applications.

2.6 Gear Design

A gear design, like other complex product design, undergoes a series of iterative revisions leading to a number of design alternatives that are evaluated with respect to the different design objectives. Especially addendum modification design of spur and helical gears involves multiple design objective evaluation and multiple constraints check such as gearing intervene check, undercutting check, and so on.

Several CAD/CAM systems have been developed and implemented to increase design productivity in the gear industry [123-126]. Most are standalone systems that adopt conventional design methods, with various design activities being carried out sequentially and without integration of the various stages. This sequential and non-intelligent approach is long and costly lead times in the design process. To obtain economic and viable design solutions for the gear design and manufacturing process, analysis, evaluation and optimisation should be carried out concurrently.

Yiu-Wing and Siang-Kok [127] described the development of an expert system for gearing design application and the detailed design calculations are performed within a

software environment that was developed using C. The selection and design of gears is governed by design specifications such as power transmission requirements, speed ratio, shaft arrangement, mechanical efficiency and quality of operation. However, the application results only provide the outline dimensions of the gear-set and do not include tooth form generation and 3-D model construction.

Aziz and Chassapis [128] developed an integrated environment for spur and helical gears design with geometry generation. The system integrated optimisation application, 2-D gear model and stress analysis. The advantage of the integrated system is that it provides an early estimation of the full stress fields during the design phase, where there is still time to make significant changes. However this system is not designed based on the distributed structure does not support to conduct design over the Internet. In order to support the cooperative design over the world based on the Internet, there is a need to make all the gear design resources including gear design experts, their design tools, and databases work collaboratively together in a platform over the Internet.

2.7 Concluding Remarks

A framework for enabling collaborative designs should allow a designer to access their favourite tools from hypermedia workspace. Today's Web technology supports coordination through provision of shared information space. However, to fully participate in a collaborative design, designers need to be able to, not only exchange data [40-42, 70-75] but also to negotiate their design intent governing the design generation. This negotiation requires a task-oriented view of the design project, rather than just the data-oriented view provided by the Web. Fortunately, the tools capable of supporting a task-oriented view can be implemented on top of the Web infrastructure using the latest information technologies, such as Java Servlets, intelligent search engines, XML, VRML (virtual reality modelling language), Java 3D, and middleware technologies (such as RMI, DCOM, CORBA and EJB). In addition, client side scripting, applets, and ActiveX controls often make significant contributions to the execution of design applications or tools. Furthermore, AI techniques, Genetic Algorithms, and so on are becoming enabling technologies in collaborative design, in

an attempt of improving efficiency and intelligence in design process. Some efforts have been devoted already to address these problems [43-61, 76-85, 88-91].

From these examples, it can be found that the latest Web technologies could be used to establish an open architecture based on the existing Web infrastructure for communication, facilitate collaborative design activities or capture the collaborative session. Most of the proposed systems are still under proof-of-the-concept prototype development stage. It is clear that challenges in these areas will remain as a research opportunity. In summary, the following areas have been identified as future research opportunities and challenges:

1. *System architecture with decision mechanism for Web-based collaborative design.* The architecture of a collaborative design system needs to be carefully formulated to make full use of the Web features to capture fuzzy information and facilitate creative design generation. From client's perspective, these features include client-side scripting, applets, ActiveX controls, DII of CORBA, XML-based response in extensive use of CGI, Java Servlets, ASP, JSP, PHP, EJB, DSI of CORBA. Special attention should be paid to deal with the limitations of Web technologies. Using Web technologies could obtain fundamental communication architecture for collaborative engineering. However, clients are not passive user but design partner.
2. *Intelligent mechanism.* Artificial intelligence mechanism should be added to improve client's side ability for making design decision. More flexible and freer design resource interaction should be facilitated in collaborative environment to support complex design making process.
3. *Scalability, openness, and heterogeneity.* The system architecture can accommodate any growth in future load such as sequent computers and tools and can be easily extended and modified. Any new components integrated in the system can communicate and work together with some of components that already exist in the system. The distributed system is constructed using different programming languages, operated on different hardware platforms and obeys different protocols. The system architecture should supports communication with heterogeneous components.

From the context of gear design domain, most of CAD/CAM systems that support gear design are standalone. Current integrated gear design systems that facilitate concurrent design structure are not designed based on the open distributed system. As an extension of concurrent engineering applied in design engineering, collaborative environment based on the Internet is needed in gear design domain, to obtain more flexible integrated design environment with the following features:

1. *Integrated design environment based on the Internet.* In gear design area, there are many existing design applications and continuing coming applications, to be integrated and communicated with each other. These systems may be executed in different operating systems such as Unix, Windows, OS2, and so on, and may be written in different programming languages such as C/C++, Java, and so on.
2. *Legacy codes reusability.* Existing applications should be integrated seamlessly together with a new application without code rewriting and can be interoperated with each other. More reusable components mean more efficient development processes, more reliable application systems and lower developing cost.
3. *Multiple working models support.* Over the Internet, Web-based design system should facilitate multiple working models supports such as multiple programs working model, singular large scale computing program invocation, design resource sharing among multiple designers, integrated design environments that invoke multiple programs, and so on.
4. *Multiple design interest balance mechanism* Gear design involves many design disciplines. Co-design among multiple disciplines needs an effective mechanism to balance multiple interests.

The combination of all the above features to provide a robust tool for Web-based collaborative environment for integrated gear design is a novel and challenging task.

The tasks in this project are to develop such a Web-enabled collaborative environment with the above-mentioned features for design and manufacture, and to apply the environment to the design and manufacture of spur and helical design, and worm gearing design systems.

Chapter 3 Overview on Architectures of Distributed System

3.1 Introduction

A collaborative design is a collection of either the co-operated efforts undertaken by a team of designers and other specialists, or the one-person-made multi-concerned decision making use of multiple criteria defined by multiple other experts. In this study, the later one is addressed and focused on. Partners perhaps diverse geographically and even across time zones, it is not always convenient to make all the partners available at any time. In addition, products are increasingly complex so that one single procedure of design process would take a long time. Thus the system design has to include the consideration of the situation that partners are absent but have their resources available, such as the domain-specific application tools, data files, or service programs. An active designer could directly access essential resources to pass design requirements, retrieve resultant data, invoke service program, or even execute remotely the specific tool to implement a design task.

Establishing such an environment has to take consideration of the following aspects:

Heterogeneous application integration – In a collaborative environment, many applications are needed for integrated design and manufacture, including different kinds of large size computing programs, CAD/CAM/CAE systems, and databases accessing services. These systems are mostly standalone and heterogeneous, e.g. written in different languages such as Java and C/C++ and running on different platforms such as Windows, Unix, OS2, or Macintosh, and required to be integrated together to implement a complex design task. A proper infrastructure is needed to provide a common interface to enable all the heterogeneous systems to be integrated together.

Communication of multiple programs: To a complex product design task, there may be a need for invocation of multiple programs during the design procedure. The proposed system should provide a communication mechanism to support such an invocation between programs, and a controlling mechanism to help to execute multiple programs logically for certain design purpose.

Legacy system reusability: System development cost is an inevitable problem to be taken into consideration. Reuse of legacy system is a methodology for reducing the high cost of software development and maintenance. It is necessary to build an environment to accommodate many existing design and manufacturing applications without rewriting their essential codes.

Large size and time-consuming program invocation -- In engineering field, there are many large-size of computing programs for product design and analysis. These programs are not convenient to work together due to its time consuming feature. System needs to provide facilities to support these programs to execute singularly.

To address these challenges a hybrid infrastructure is considered for this study. In this chapter, two basic architectures of distributed systems, i.e. point-to-point structure and server-centralised structure, and their benefits and drawbacks are firstly introduced in Section 3.2. Point-to-point structure and its implementation technology are then described in Section 3.3. In Section 3.4, server-centralised structure is illustrated. Hybrid architecture of these two structures is presented in Section 3.5 and combination of multiple technologies are considered and described in Section 3.6. At last it ends with the summary part.

3.2 Fundamental Architectures of Distributed System

In a collaborative environment, multiple applications need to work together. From a user's point of view it would be nice to have one system that can integrate all the applications as needed and that provides a unified graphical user interface.

From a resource owner's perspective, the application should be ported and executed on the original machine. Only essential information such as input parameters, output parameters and application execution command is sent to the remote user, other than the essential codes.

From a programmer's perspective, an integrated architecture requires a tight association between the algorithm implementation of an application and its graphical user interface, and also allows them to be developed using the developer's favourite languages and running on their own platforms. To satisfy the needs of both the user and the programmer, a flexible middleware layer has to be introduced in place, which separates the client from the application functionality and allows utilisation of applications within a single graphical user interface.

A distributed architecture can be considered to fulfil such a strict separation of algorithmic functionality i.e. design programs and graphical user interface. Currently, there are basically two types of architectures possible to realise this distributed platform [129]. In a three-tier configuration as shown in Figure 3.1, a central Web server takes on the task of a mediator and brings together application providers and interested clients. It provides administrative services and controls all interactions.

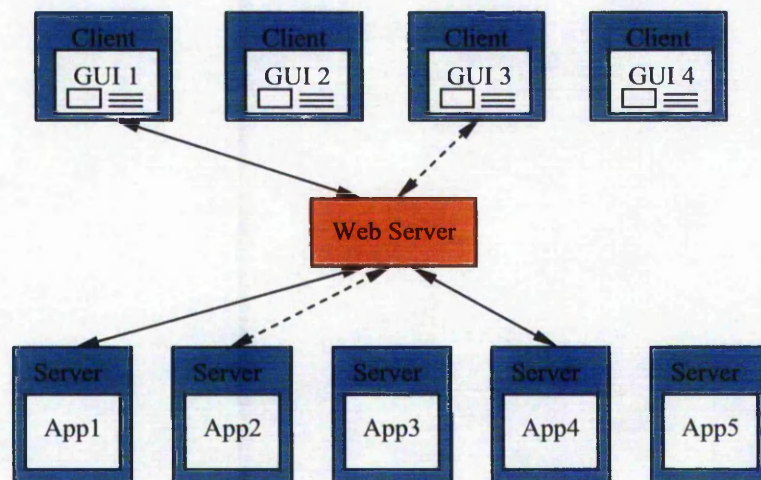


Figure 3.1 Web server centralised architecture

On the other hand, a totally decentralised system could be conceivable, too. Clients could contact application servers without going through a central station, as shown in

Figure 3.2. This point-to-point architecture would avoid a central bottleneck, however, each application component server would have to implement its own administrative system.

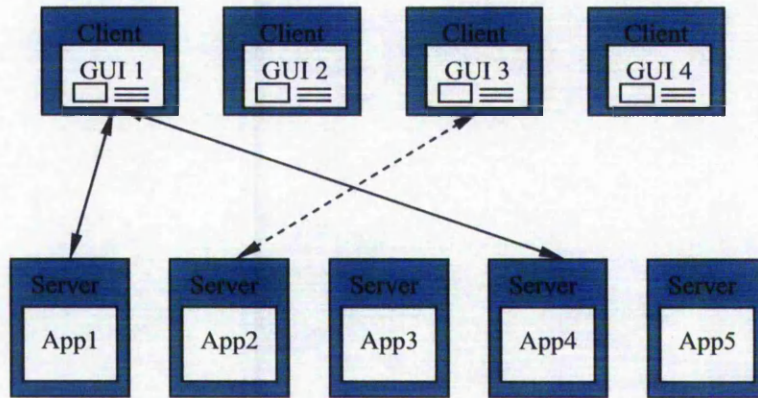


Figure 3.2 Point-to-point architecture

In this study, these two structures are used in different paradigms in the integrated design environment. Point-to-point direct connection structure is used in the circumstances with multiple programs integration to make sure higher efficiency due to the direct communication. Server-centralised model is used for the remote execution of large size program, as seen in details in chapter 6. In addition, hybrid structure of these two structures is also used in application paradigm demonstrated in Chapter 7, in order to obtain direct functional communication with high performance and convenient administration based on Web server.

3.3 Point-to-Point Communication Structure

3.3.1 CORBA-Based Point-to-Point Structure

CORBA facilitates the development of distributed systems by enabling transparent access to the distributed objects on different locations, written in different languages, on different operating systems. It is based on client/server architecture and can support the direct communication in point-to-point model. A client is a process that wishes to perform an operation on a distributed object. A server is a process that provides this object to the client. CORBA enables the client to transparently invoke operations on such distributed objects without any regard of their physical location, the programming

languages they were written in, or the operating system they were running on. The client accesses such objects in the same manner as if they would reside on its own machine.

The main part of CORBA is the Object Request Broker – ORB. The ORB is the middleware that establishes the client-server relationships between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across the network, as shown in Figure 3.3. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming languages, its operating system, or any other system aspects that are not part of an object's interface. In so doing, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

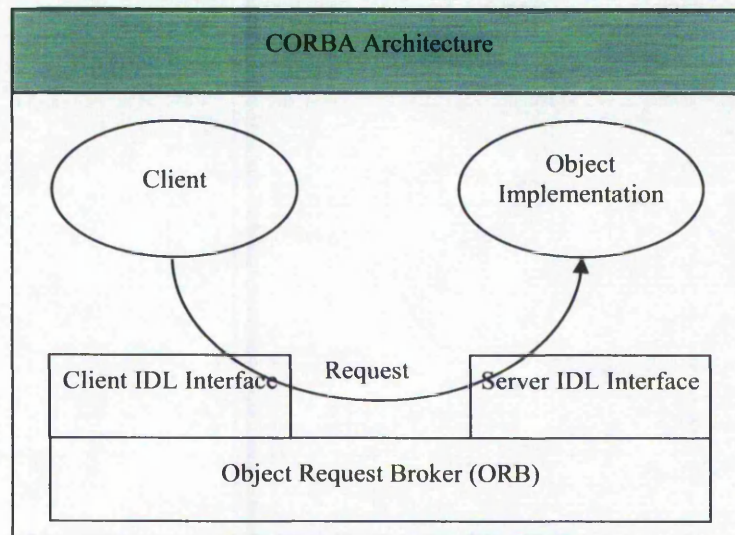


Figure 3.3 CORBA based point-to-point communication

Client is the process that wishes to perform an operation on an object and the Object Implementation is the code and data that actually implements the object (the object implementation resides on the server). The ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare it to

receive the request, and to communicate the data making up the request. The interface the client sees is completely independent of where the object is located, what programming language it is implemented in, or any other aspect that is not reflected in the object's interface. The client does not know anything about the object's internal structure or how the object is implemented; it only knows its interface. This describes the basic principle of CORBA: the separation of object's interface from its implementation.

In CORBA, an object's interface is defined with a single platform independent language – the Interface Definition Language (IDL). This interface definition is independent of the object's implementation, and programmers can choose the most appropriate operating system, execution environment and programming language for each object (component of the system). This enables the object owners keep their favourites to the language, platform and operating systems, protect their intellectual property and only offer the objects invocation. On the other hand this also makes the client-side developers not bothering to enter the object implementation details. Moreover, CORBA also allows the integration of existing components into a distributed system; the programmers just have to write some wrapper code that translates between the old application interfaces and the CORBA interface.

The main features of CORBA are:

- ORB Core
- OMG Interface Definition Language (OMG IDL)
- Interface Repository
- Language Mapping
- Stubs and Skeletons
- Dynamic Invocation and Dispatch
- Object Adapters
- Inter-ORB Protocols

Most of these are illustrated in Figure 3.4, which shows the components of CORBA relate to one another. Each component is described in detail in appendix B

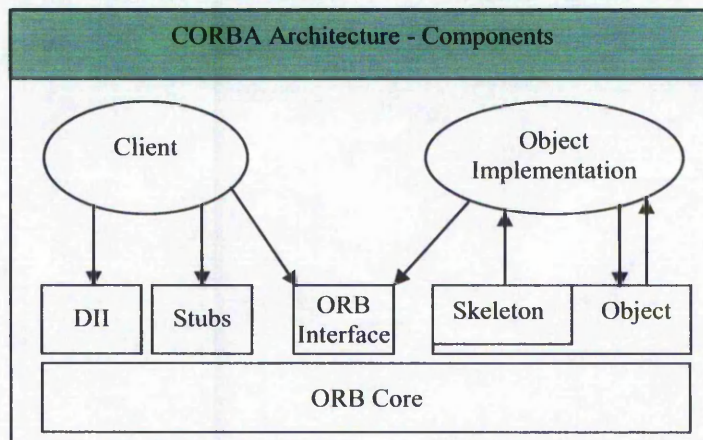


Figure 3.4 Common Object Request Broker Architecture

3.3.2 Comparison of CORBA and Other Distributed Technologies

Some distributing tools have certain functions similar to CORBA such as, Remote Procedure Calls (RPC), Distributed Component Object Model (DCOM), and Java Remote Method Invocation (RMI). However, despite certain progress, they are rather limited in comparison to CORBA as summarised below:

- RPC:
 - Does not provide object abstractions, message passing, or dynamic invocation
 - Does not address inheritance of interfaces
- DCOM:
 - Platform dependent (Windows)
 - Does not support heterogeneous distributed computing
- Java RMI:
 - Language dependent (Java)

Web Services technique is an emerging distributed middleware in addition to CORBA. The communication protocol for building Web Services is known as the Simple Object Access Protocol (SOAP). SOAP is a lightweight, XML-based, and platform and language independent protocol that allows applications to exchange structured information across the Web, usually over HTTP. SOAP consists of three parts: an

envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP codifies the existing practice of using XML and HTTP as a method-invocation mechanism [113].

Compared with CORBA, Web Services/SOAP has both advantages and limitations due to some of its import characteristics (+ represents advantages, - represents limitations):

- It uses XML-encoded plain text to send data.
 - + More widely adopted
 - + Easier for developers to read and debug
 - Large message size requires much more network bandwidth
 - Complex XML parsing and marshalling needs more CPU time
- It uses HTTP protocol for communication.
 - + Can pass through firewalls
- Its interface is Web Services Description Language (WSDL).
 - Lower level interface description language
 - More difficult to create and understand for humans (compared to CORBA/IDL)
- It is a simple and lightweight protocol.
 - No standardised transaction and security context

CORBA is not a markup language like SOAP. CORBA works as a go-between, making the connection and communication between heterogeneous distributed and object-oriented applications possible. CORBA offers a solid and comprehensive development platform for large-scale projects in which managing complexity and ensuring reliability are major concerns. It also ensures excellent transaction rates and fail-over capabilities. Although CORBA is powerful, it is difficult to develop and deploy. The choice of CORBA for small projects may not justify its performance benefits because it requires considerable code development.

In contrast, SOAP is built from two simple and well-known standards, HTTP and XML, and is furnished with the smallest possible XML vocabulary to package the essential data for RPC. It is easy to learn and implement. The use of the two textual protocols makes network data transfer heavy, requiring more bandwidth, but testing and debugging become easier.

SOAP is wire protocol. SOAP lacks the activation elements, security, and state management that CORBA provides. The XML data inside the SOAP envelope must be extracted and parsed, thereby degrading the performance. In comparison to CORBA's binary data, SOAP's text-based data consume significantly more bandwidth.

Finally, the Object Management Group has taken important steps to ensure compatibility between CORBA and new communication protocols. There are a number of bridges, including open-source projects, capable of translating SOAP requests into CORBA invocations. Thus, by utilizing SOAP, CORBA can still remain as the first choice for large-scale projects where performance and reliability are critically important.

3.4 CORBA-Based Server-Centralised Infrastructure

The server-centralised distributed environment, consisting of three tiers: User tier, Web server tier, and back-end database tier, as shown in Figure 3.5.

- (1) User tier — Valid customers of the environment are allowed to visit and load the HTML pages, with Java applet, located on the Web server tier over the Internet. On the user side, a Java plug-in Web browser is enough and what the user needs to know is just how to operate the graphic interface, and does not have to be aware of any CORBA technical details behind it.
- (2) Web server tier — This consists of HTTP server used as the collaborating server and CORBA object server. From a common HTML page, CORBA objects could not be visited. Java applet can include IDL-generated client stubs, which let it invoke objects on the CORBA server. Through HTTP protocol and the linkage mechanism between HTTP Server and CORBA Server application objects can be invoked from the HTML page. CORBA ORBs are in charge

of linking and finding the essential server side CORBA application objects. All these applications are not only dispersed geographically and also written in different languages, working on different platforms and operating systems, and encapsulated into components that CORBA service facilities can find. The interface of each application is separated from the implementation written by the respective language the partner chooses to use. All these applications consist of rich resources for conducting collaborative design tasks such as different computing programs, various kinds of PDM (Product Data Management), CAD and CAM software, accessing and searching applications for back-end database resources, and other resources from customer services.

- (3) Back-end database tier — This tier is a database server that gathers and stores historical and real-time data received in the process of design and manufacture. The databases can be of different types such as JDBC, DBMS, SQL, etc.

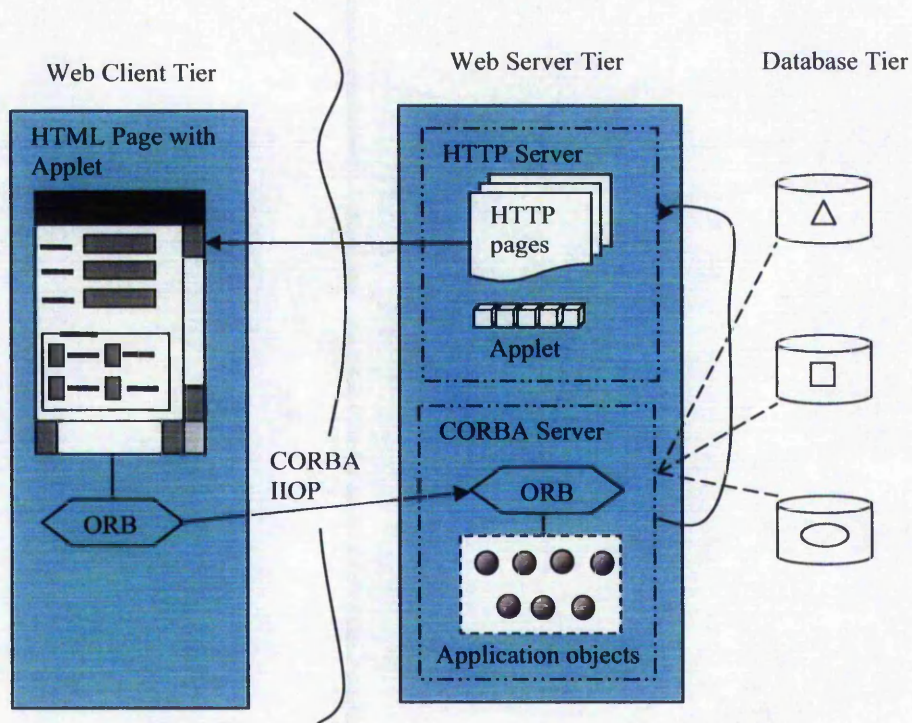


Figure 3.5 Architecture for the Internet based design environment

Such a server-centralised structure is used for remote invocation of singular large-scale program as presented in details in Chapter 6 and for on-line worm design from browser as presented in details in Chapter 7.

3.5 Hybrid Structure

The Web-based design architecture developed by this research is mainly based on the concept of combining a Web-server-centralised system and peer-to-peer approach. Figure 3.6 illustrates the architecture consisting of four types of components. There exists at least one Web server, i.e., the collaborating server, which acts as a central unit and manages the interaction between users on the one side and application providers (e.g. servers) on the other side. Design services such as computing, and analysis tools are provided by numbers of object servers, which form the functional basis of the distributed network design system. Client procedures allow users to access available design applications and thus, utilise the tool's functionality. Finally a Naming Server is employed so components can identify each other's locations.

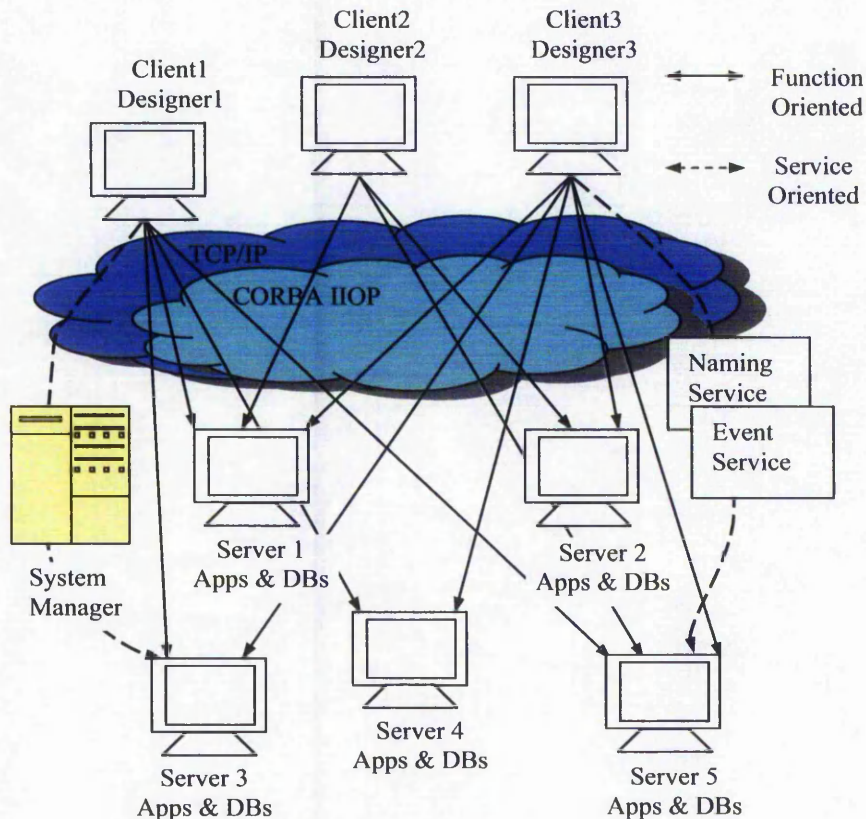


Figure 3.6 The hybrid architecture

The communication between all components is based on the CORBA middleware system with its transport protocol IIOP (the Internet Inter-ORB Protocol). In the CORBA-based distributed system, existing resources are encapsulated into objects, e.g. components that can be found and invoked between each other. IIOP an underlying mechanism of CORBA, is the standard protocol that specifies how objects communicate across TCP/IP (Transmission Control Protocol/Internet Protocol), the standard connection-oriented transport protocol for the Internet. In the client/server distributed architecture, a client is a process that wishes to perform an operation on a distributed object, and while a server, an object provider, hosts a process that provides this object to the client. The system allows a client to find, further connect and invoke the selected object that a server hosts. With the standard interface sequent new objects and new clients could work along with the already existing components. As the system augments there may be a need of one or more managers to perform the collaboration tasks. A collaborating server provides a set of server-centralised services such as user authentication, object selection, and data-related operations.

Moreover, the CORBA facilities such as Name service and Event service are utilised so components can be located and found easily.

This structure is used in distributed design optimisation application and presented in Chapter 4 and Chapter 5.

3.6 Encapsulation of Legacy Applications

There are many existing heterogeneous legacy applications used for engineering design and product development. Turning these applications into distributed components to make them reusable is an effective way to reduce development costs. CORBA eases the transition from standalone legacy applications to more flexible object-oriented distributed components. Here in this project the *legacy software applications* refer to those application software designed and implemented with traditional technologies (that is, without distributed concepts and object-oriented system development technology), that are still valuable to perform crucial work, and

usually represent a significant investment and years of accumulated experience and knowledge.

According to Sneed [5], there may be three strategies for encapsulating legacy systems, e.g. redevelopment, reengineering and wrapping. The redevelopment strategy is to start from scratch and redevelop all of the applications with the distributed object concepts. This approach frees the developers from any consideration of the existing systems. But every function must be re-implemented and tested in a new language and in a new environment, which is expensive and time consuming.

The reengineering strategy is to convert the existing programs to object-oriented programs and distributed objects appropriately. This approach is a promising method since it is not necessary to re-implement functions whose functionalities are the same as the functionalities of the older systems. However, code conversion is not easy and few tools and methods are available.

The third strategy is to wrap components of the existing systems so that they can be invoked from the distributed environment by object-oriented method. Wrapping is a method of encapsulation that provides clients with well-known interfaces for accessing server applications or components. The advantage of wrapping is that legacy systems become part of the new generation of applications without discarding the value of the legacy applications. Wrapping is a compromise approach. The construction of object-oriented distributed systems with the first or second approach is maybe the ultimate goal. But the explosive increasing demand for applications based on Internet technology and object-oriented distributed environments does not permit unpredictable time delay. Wrapping is a realistic approach, since it can be accomplished easily and rapidly with current technologies.

To use the wrapping techniques well, application developers must have a good understanding of the wrapping structure and implement the interfacing techniques to the legacy systems. The wrapping structure based on CORBA model is described in Section 4.3.

3.7 Combination of Java with CORBA

Java is an object-oriented programming language developed by Sun Microsystems. Since its introduction, Java has quickly become a standard language for writing Internet applications. Java language was designed to be small, simple, and portable across platforms and operating systems, both at the source and at the binary level, which means that Java programs (applets and applications) can run on any machine that has the Java virtual machine installed.

Java is usually mentioned in the context of the World Wide Web, where browsers such as Netscape's Navigator and Microsoft's Internet Explorer claim to be "Java enabled". Java enabled means the browser can download and play Java programs, called applets, on the clients' system. Applets appear in a Web page much the same way as images do, but unlike images, applets are dynamic and interactive. Applets can be used to create animation, figures, forms that immediately respond to input from the reader, games, or other interactive effects on the same Web pages among the text and graphics.

3.7.1 Platform Independence

Platform independence -- that is, the ability of a program to move easily from one computer system to another -- is one of the most significant advantages that Java has over other programming languages, such as C++, particularly if software needs to run on many different platforms. In the development for the collaborative design systems over the World Wide Web, it is crucial to be able to run the same program, for example, the client-side applications, on many different systems. Java programs can run on any system for which a Java virtual machine has been installed. This is given in details in Appendix C.

3.7.2 Java Servlets

A servlet is a small piece of Java code that a Web server loads to handle client requests. Unlike a CGI application, the servlet code stays resident in memory when the request terminates. In addition, a servlet can connect to a database when it is initialised and then retain its connection across request. Servlets do not provide the full power of distributed object interfaces, however, it is precisely this constraint that convenient for

simple applications. For example servlets make good replacements for CGI-bin scripts, used as Web server extension. In the server-centralised applications they are good at accepting from input, interacting with a single database, and then dynamically generating an HTML response page. So there may still be a place for servlets in the Java server-side arsenal. In this project, servlet-based application is used for invocation of single large size of program.

3.7.3 CORBA and Java

Java itself is not client/server oriented, nor is writing client/server applications in Java easier than for example in C++. Introducing CORBA to the Java environment means that Java applets are no longer restricted to simple interaction with the user, but are instead capable of taking part in complex interactions with backend services. A combination of the Java programming language with the CORBA standard for application integration presents a good solution for application components capable of accessing multiple, shared backend services located across the Internet.

CORBA provides network transparency and Java provides implementation transparency. CORBA reciprocates by enabling Java to interface with different programming languages and extends Java by providing a framework for distributed object communications. CORBA provides an intergalactic distributed object infrastructure over the Internet, and guarantees inter-operating ability between vendors, which is essential for Internet applications, where different sites' objects, implementations, need to be able to communicate with each other.

CORBA also benefits from the combination of Java. Java simplifies code distribution in large CORBA systems—its bytecodes make it possible to ship object behaviour around. Java enables CORBA clients to be easily distributed to remote machines, regardless of platform, via applets and browsers. Java is an ideal language for writing the client and server CORBA objects. Java's built-in multithreading, garbage collection and error management make it easier to write robust networked objects.

The combination of Java and CORBA provides a robust platform for Internet based applications.

3.8 Summary

There have been numbers of existing product design applications in engineering, which lead in the challenges of development of effective integrative environments. This research utilises the CORBA technology and Java to address the challenges.

Standard interfaces CORBA provides are important to develop sharable applications over heterogeneous systems. CORBA not only provides the direct communication mechanism in peer-to-peer between heterogeneous applications, but also supports plug-in working model from web pages based on the server-centralised model.

In the direct point-to-point architecture clients could contact application servers without going through a central station and thus it would avoid a central bottleneck. This leads to the effective communication structure. On the other hand, server-centralised system would provide administrative services and controls all interaction, where the server acts as a mediator to bring together application all providers and interested clients. In this research, these two architectures are used in different application paradigms. Multiple structures are used in the integrated environment in order to provide a more flexible framework for different working models. Especially in the open distributed system to support flexible design collaboration over the Internet, hybrid architecture of multiple structures is applied to provide the powerful environment to facilitate different application integration requirements.

Java provides the client side GUI developing tools and server-centralised server extension tools to enhance the platform-dependent functions in distributed systems. Java applets are not only capable of simple interaction with the user, but also capable of taking part in complex interactions with backend services in CORBA-based distributed system. A combination of Java with CORBA for application integration presents a good solution for application components capable of accessing multiple, sharing backend services located across the Internet.

CORBA and Java are complementary to ensure high source portability and robust platforms for Web-based applications.

For the knowledge gap between distributed technology and design resources, CORBA technology provides seamless access to distributed computation and data resources, and CORBA-based collaborative environment systems facilitate distributed collaboration and reuse of design resources.

From the perspective of gear design, the hybrid CORBA based framework will bridge the knowledge gap where heterogeneous gear design applications and tools needs to be integrated and communicated with each other for the integrated gear design in a distributed way over the Internet.

Chapter 4 Distributed Gear Design Optimisation

4.1 Introduction

Globalisation has resulted in more distributed, decentralised design resources including design teams and computer-aided software. A product design, such as gear design, needs to be implemented using diverse design tools by multiple design members via the Internet. Collaborative engineering enables multiple specific domain experts and their computerised software tools to work together to search one design space and obtain a common solution. However, most design problems are difficult and complex, and affections of the models from different specialists to the design solution are comprehensive. Therefore, in the collaborative design environment, a resolution strategy to such a complex situation is inevitable for an active designer (or main designer) to leverage multiple interests.

The optimal or rational design of gears is a good example to study such kinds of problems. As described in Appendix C, gear design includes a large number of variables, limitations and performance objectives. The compromise between the various objectives is not easy to find because the optimisation criteria are often contradictory. In addition, the design space is massive and multi-dimensional and it is impossible to manually generate and examine every potential gear design. For example, 10 values of each of the four design parameters: pressure angle, module, tooth number, and profile modification coefficient can result in as many as 10,000 candidate designs. Furthermore, majority of these variables take discrete values, hence it is not easy to fully optimise a design using traditional optimisation methods, which mostly need the implementation of a gradient. Moreover, the problem of the optimisation of gears, like many other engineering designs, admits many solutions and often the one selected is the one that adapts best to a given environment.

Genetic algorithm (GA) is an ideal solution to the gear design optimisation. The nature of GA meets the requirements in this research in two aspects. Firstly, GA does not have much mathematical requirement about the optimisation problems and can handle any kind of objective functions and any kind of constraints (i.e. linear or nonlinear) defined on discrete, continuous, or mixed search spaces. It would accommodate any complex consideration from multiple domains. On the other hand, the ergodicity of evolution operators makes genetic algorithms very effective at performing global search (in probability). It would benefit from a broad region of search and retrieve a wide class of alternative design solutions [100]. The GA approach and relevant optimisation technique details are presented in Chapter 6.

Affections of different specialists' specific knowledge on the design results are formulated as optimisation objectives or constraints. In the optimisation procedure, calculations or analyses related to these pieces of specific knowledge would be implemented as needed. In the computer-aided design world, they might be included in some forms of resources such as pieces of program, autonomous CAD software, knowledge database, or a database file.

Many engineering tools, such as CAD tools, calculation software tools, engineering databases and knowledge-based systems, which have been used in engineering and are still playing important roles, are mostly standalone and completely autonomous. Therefore it is important to develop a proper mechanism enabling invocation between design resources over the Internet.

CORBA is utilised to develop a distributed framework to integrate all the design applications and implement the interoperation between these distributed and heterogeneous applications. In order to help designers to conduct collaborative design among multiple experts, a design optimisation mechanism, i.e. optimiser is needed to balance multiple design interests to achieve rational or optimal solutions.

This chapter is organised as follows. Section 5.2 describes the development, integration and communication of distributed components. Section 5.3 presents the

distributed system architecture of the gear design optimisation, stressing the Web technologies used for this system. Section 5.3 introduces the working procedure of distributed gear design optimisation. Section 5.4 discusses automatic gear tooth generation. Section 5.5 is about implementation of GUI on Windows. Section 5.6 gives implementation of evaluation programs as CORBA application object on Linux. Section 5.7 presents implementation of algorithm application as CORBA client on Windows. The chapter ends with summary Section 5.8.

4.2 Development, Integration and Communication of Distributed Components

4.2.1 Wrapping Structure Based on CORBA

The major purpose of this research is to enable open interconnection of a wide variety of languages, implementations, and platforms by utilising CORBA technology. CORBA is the technical solution for building an integrated design system over the WWW.

Software development in CORBA environments is performed at two sides: client-side and server-side. The server-side application developer implements object implementations for services, and then describes the interfaces to the provided services with IDL (Interface Definition Language). A client-side application developer makes reference to the IDL written by a server-side developer, and implements client applications. The interface of a service (or an object) is always separated from the implementation.

The wrapping structure using CORBA technology is shown in Figure 4.1. It hides differences in programming language, location, and operating system with the interface defined in IDL (Interface Definition Language). Server-side application developers only need to understand the services of legacy systems and to describe them in the interface with IDL. A legacy application can be partitioned and componentised to facilitate to CORBA based distributed environments. Each component can be encapsulated separately, and then integrated together by using object-based communications.

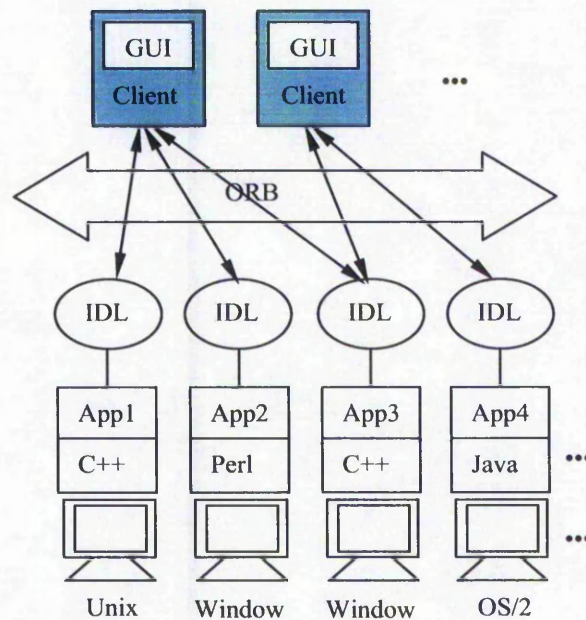


Figure 4.1 Encapsulating structure

The IDL is used to define interfaces in CORBA. An IDL interface file describes the data types and methods or operations that a server provides for an implementation of a given object. IDL is not a programming language; it describes only interfaces and it has no implementation-related construction. The CORBA does specify mapping from IDL to various programming languages, including C, C++, Smalltalk Ada, COBOL and Java. IDL is used to show interfaces to many CORBA objects. In this research IDL is used to show interfaces for the Java and C++ objects.

4.2.2 Developing Procedure of Distributed Components

Taking the advantages of CORBA, the distributed applications within the system can be inter-operated between applications, regardless of what language they are written in or where they reside on. In this project, communication of Java-to-Java, C++-to-C++, Java-to-C++, Linux -to-Windows are implemented based on CORBA.

Figure 4.2 illustrates the whole procedure from IDL to the establishment of final client/server structure. A server side application written in C++ is ported on a Linux system, while a client side program written in Java locates on Windows system. Both sides communicate with each other through IIOP protocol. The implementation procedures are described as below.

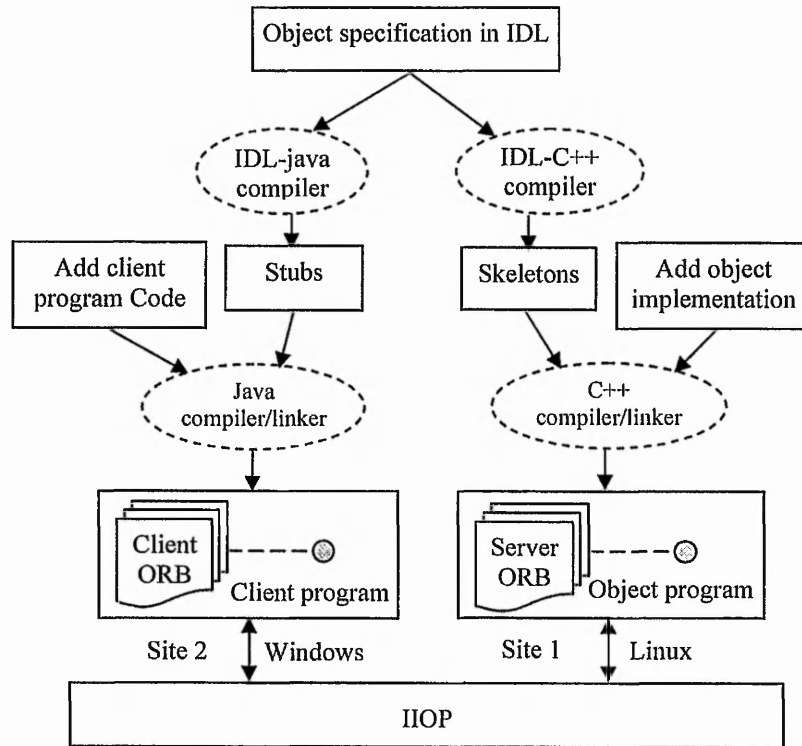


Figure 4.2 Procedure from IDL to the establishment of final client/server structure

4.2.2.1 Definition of Object Interface

The first step to create a distributed application with CORBA is to specify all of objects and their interfaces using the Interface Definition Language (IDL). The IDL can be mapped to a variety of programming languages, such as C++ and Java as shown in this example.

Taking a gear stress calculation package as an example, Figure 4.3 shows how to define the interface by IDL for an existing application.

The lower part of Figure 4.3 is the IDL definition of interface for the *GearStress*. Three interfaces: *inputdata()*, *main_function()* and *outputdata()* are mappings of the three operations: data input user interface, the main calculation program and the results output user interface in the standalone gear stress calculation package respectively.

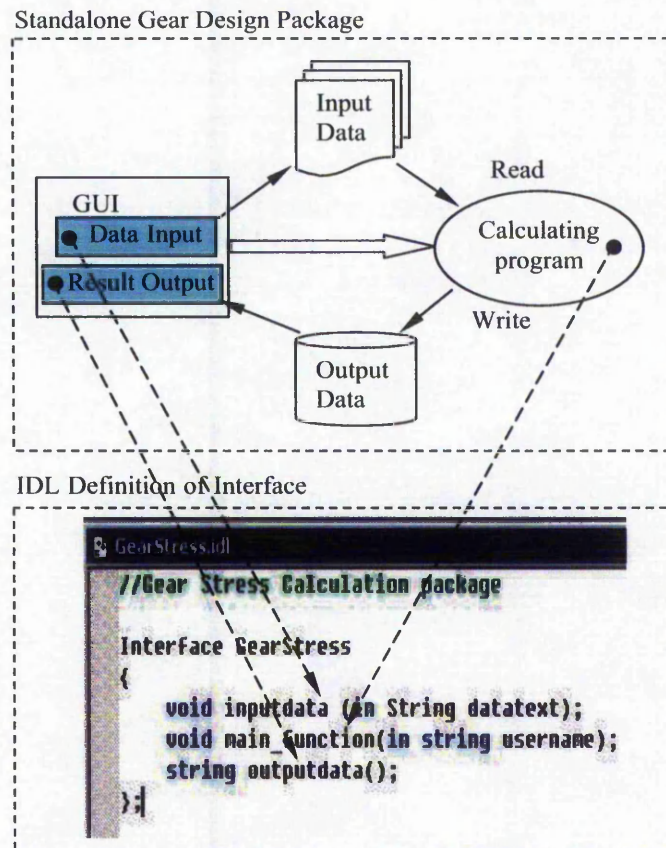


Figure 4.3 Interface definitions for the existing stress application

4.2.2.2 Generating Client Stubs and Server Skeletons

The server skeletons and client stubs are generated from the IDL interface definition file by using appropriate compilers. The compiler type depends on what kind of programming language to be used for the CORBA object development. Corresponding to C++ and Java, the two most popular programming languages used for CORBA object development, *idl-to-cpp* and *idl-to-java* are the two most popular compilers used for generating stubs and skeletons. Considering the situation in Figure 4.2, an *idl-to-cpp* compiler is used to produce C++ skeletons, e.g. server-side ORB files, while the Java stubs, e.g. client-side ORB files, are generated by an *idl-to-java* compiler.

4.2.2.3 Implementing the Client

A client side developer needs to write a GUI-oriented client program for a user to invoke remote objects. The client program, *Client.java* written in Java, is designed for

implementing the client side tasks, including passing the parameters, invoking a design program, and retrieving the results. In order to implement these procedures, the client program performs the following tasks:

1. Initialising the ORB.
2. Binding to a *Geardesign* object.
3. Invoking *inputdata()*, *main_function()* and *outputdata()* on the *Geardesign* object.

4.2.2.4 Developing the Server and the Object Implementation

Just as with the client, many of the classes used in implementing the server are contained in the server side header files generated by the *idl-to-cpp* compiler.

On the Linux side, the *Server.C* file is a server implementation, which normally the server-side programmer would create. This file implements the *Server* class for the server side. The server program does the following tasks:

1. Initialises the Object Request Broker.
2. Creates a Portable Object Adapter with the required policies.
3. Creates the *Geardesign* servant objects.
4. Activates the servant objects.
5. Activates the POA manager (and the POA).
6. Waits for incoming requests.

The *Geardesign.C* is the object implementation program. The essential codes in the existing programs are placed in this program.

4.2.2.5 Building

Object implementation program along with its server-side ORB files and client program along with its client-side ORB files are built respectively in their own specific development environment.

Once the built object program is activated and gets ready to be called, a client side program could invoke the server-side program through ORB routines and IIOP protocol, as though invoking a method of the program itself.

4.2.3 Integration Model of Distributed Objects

4.2.3.1 Different Formats of Java Clients Sharing an Object

In engineering design practice, it is quite a common requirement for designers to invoke and share a remote service (e.g. an object) within their own programs. These programs may be designed in different program languages. Even written in the same program language, such as Java, the programs may exist in different forms. Figure 4.4 shows a distributed model based on multi-clients sharing an object.

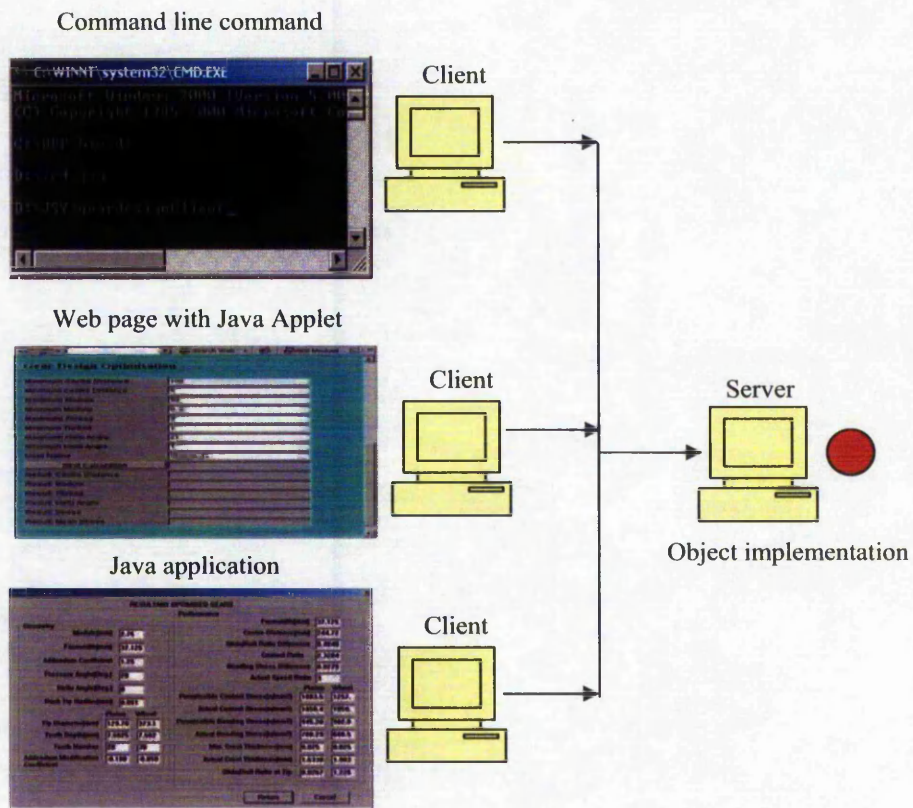


Figure 4.4 Different clients sharing an object

Every client program is designed for a certain purpose with a user's favourite format. The client routine may be designed respectively as command-line execution, plug-in

Java Applet and Java application. In this research, all the three types of client programs are applied to explore more possibilities for providing more flexible client program developing models. The clients may also be a VC++, or Python program.

A command line program or a Java application can be implemented by the developing procedures as described in Section 4.3. However, in the case of Java Applet, where the Applets are inserted in the Web page on the server, and downloaded to the client side machine, additional considerations have to be taken into account. Java sandbox security prevents unsigned Java applets from communicating with other servers except for the Web server from which the applets were downloaded. A special technique is investigated and applied to solve the problem. Chapter 8 provides more details on this issue.

With regard to the command-line format, client program, as described in Section 4.3.3, itself includes CORBA-specific routines such as ORB initialisation and binding or reference to a specific remote object, and invoking the methods on the object just as invoking local functions.

With regard to the Java application, it has a common routine e.g. Graphical User Interface in addition to the CORBA-specific routine. The CORBA client either may be handled into a process added in the common routine GUI program or included in it. The former is easier and can make CORBA-related development separated from common programming. However the efficiency of execution will be reduced.

Any format of Java program can be ported and executed on any platform and operating system since it is platform-independent. The client program, linking to certain remote objects with certain purpose, can be pre-developed, ported on Web server and downloaded when a user needs it. CORBA ORB programs have to be downloaded together onto the user machine, for the command-line program and Java application GUI. With the Java Applet, the user machine does not need any setup for CORBA in advance.

4.2.3.2 A Client Application Invoke Multiple Objects

A purpose of collaborative design is to enhance the designer's individual ability and thus a designer could need to invoke multiple remote objects to co-making design. Figure 4.5 illustrates how one common client GUI program to invoke multiple objects.

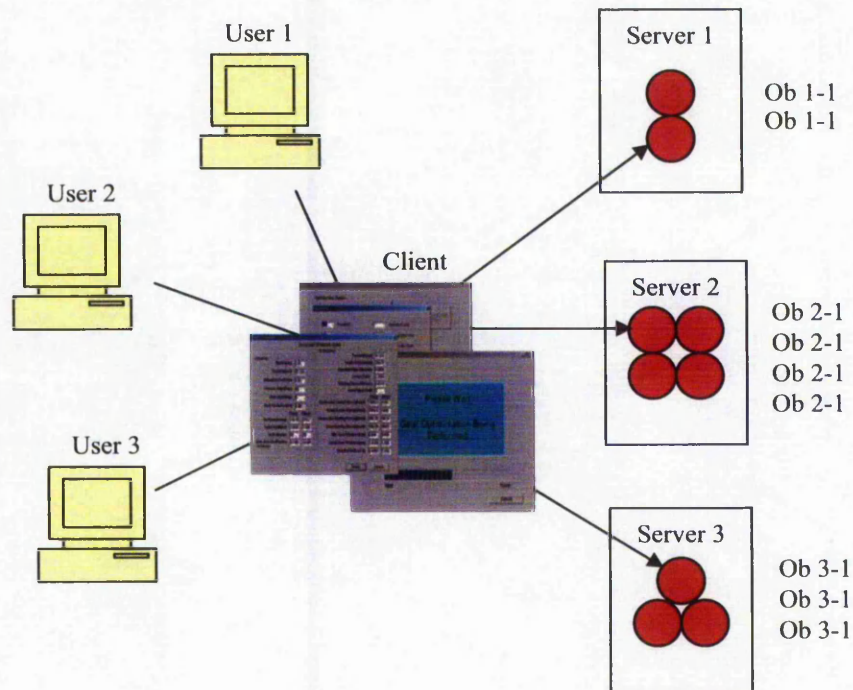


Figure 4.5 One common client GUI program invoking multiple objects

The objects may be remotely located in different computers with different operating systems such as windows, UNIX, or OS2, and written in different languages, such as C++, Python or Perl. The client is designed for implementing one single complex design task that needs to invoke multiple objects to join in.

In this project, a multiple objective design optimisation program is linked to multiple remote objects, which are a set of evaluation functions for design solutions and separately ported on Linux and Windows operating systems. During one design procedure, the remote objects can be invoked thousands of times. The user of client program does not need to be aware of any details about remote invocation. It seems that a local program is invoking local functions.

4.2.4 Management of Objects

There might be many client programs, and objects and their CORBA servers integrated in the distributed system. Methods for a client program to find the required object based on CORBA are presented in this section.

4.2.4.1 The Naming Service

The CORBA Naming Service is utilised in the system to help to find objects by name. It is the principal mechanism for objects on an ORB to locate other objects. Names are humanly recognizable values that identify CORBA objects. The naming service maps these names to object references. A name-to-object association is called a *name binding*. A naming context is a namespace in which an object's name is unique. Every object has a unique reference, and it is necessary to define the name relative to its naming context.

A program or an application is defined as a CORBA object in the system, which can be referenced using a sequence of names that form a hierarchical naming tree, as shown in Figure 4.6.

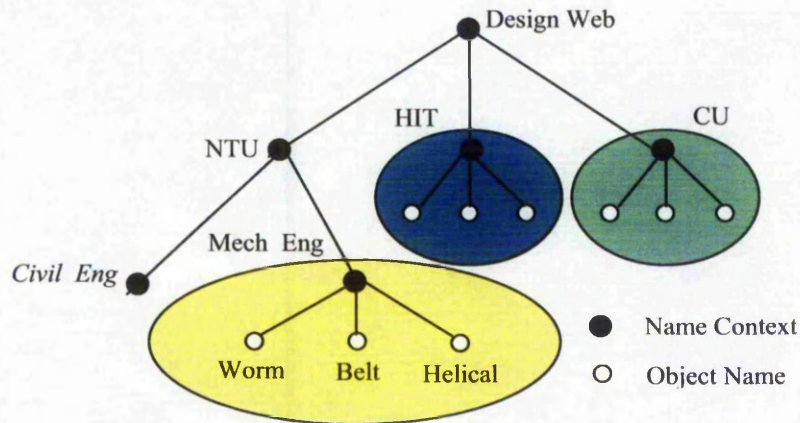


Figure 4.6 Objects hierarchical naming

In the figure, each dark node is a naming context in the naming space. An object's name consists of a sequence of names (or components) that form a context name. The last component is the object's simple name. For example, the whole name of the object Helical is *Design Web/NTU/Mech Eng/Helical*, where *Design Web/NTU/Mech Eng*

represents for the context of the Design Web. Helical represents for the name of distributed helical and spur design application objects. NTU represents for the Nottingham Trent University, HIT for Harbin Institute of Technology and CU for Chongqing University.

A client obtains information about the object it seeks, through using the object references in the naming service. An object reference uniquely identifies a local or remote object instance. Figure 4.7 shows how clients and servers interact with a name server, and how an ORB enables a client to invoke on a remote object through the naming server, which is described as below.

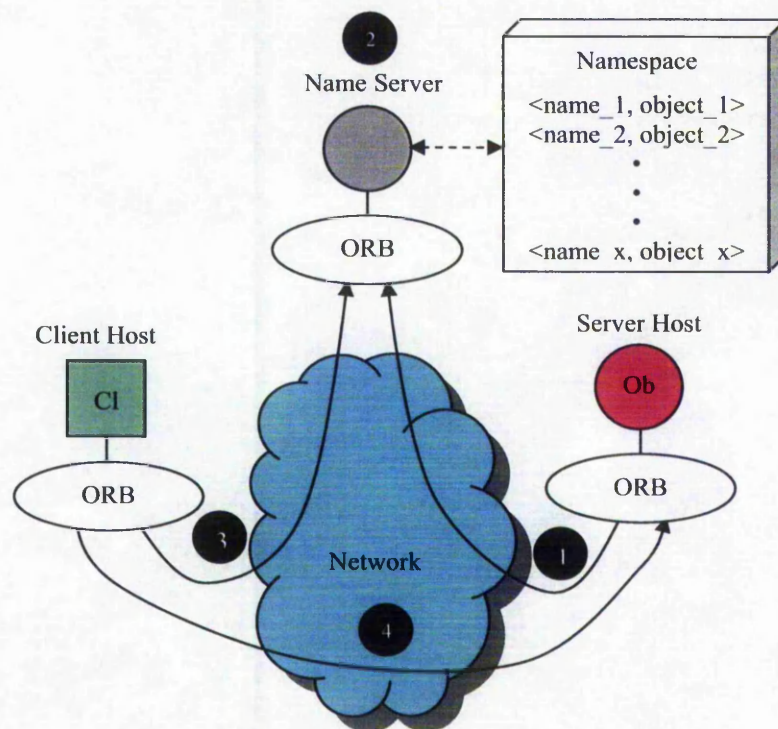


Figure 4.7 Clients and servers interact with a name server

1. When a server starts, it creates one or more objects and publishes their object references in a naming service. A naming service uses simple names to make object references accessible to prospective clients. The object server invokes function *bind (name, obj_ref)* to associate a logical name with an object reference.
2. The Name Server adds this *obj_ref / name* binding to its namespace database.

3. The client application invokes *resolve (name)* to look up the object reference by this name in the namespace. The naming service returns the server's object reference.
4. The client uses the object reference to invoke methods on the target object.

The Name Server services both clients and servers. Servers export *name/object* bindings to the Name Server; client then find these objects.

4.2.4.2 Portable Object Adapter (POA)

The Portable Object Adapter (POA) is another approach to manage access to server objects. A POA is the intermediary between the implementation of an object and the ORB and maps object references to their concrete implementations on the server, or servants. Given a client request for an object, a POA can invoke the referenced object locally.

A POA can divide large sets of objects into smaller, more manageable subsets. It can also group related objects together. For example, in a design application, one POA might handle computing objects, while another POA handles analysis objects. Figure 4.8 shows how the POA connects a client to a target object. In this instance, the server has two POAs that each manages a different set of objects.

Servers differentiate between several POAs assigning them unique names within the application. The object reference published by the server contains the complete or fully qualified POA name and the object's ID. The client request embeds the POA name and object ID taken from the published object reference. The server then uses the POA name to invoke the correct POA. The POA uses the object ID to invoke the desired object, if it exists on the server.

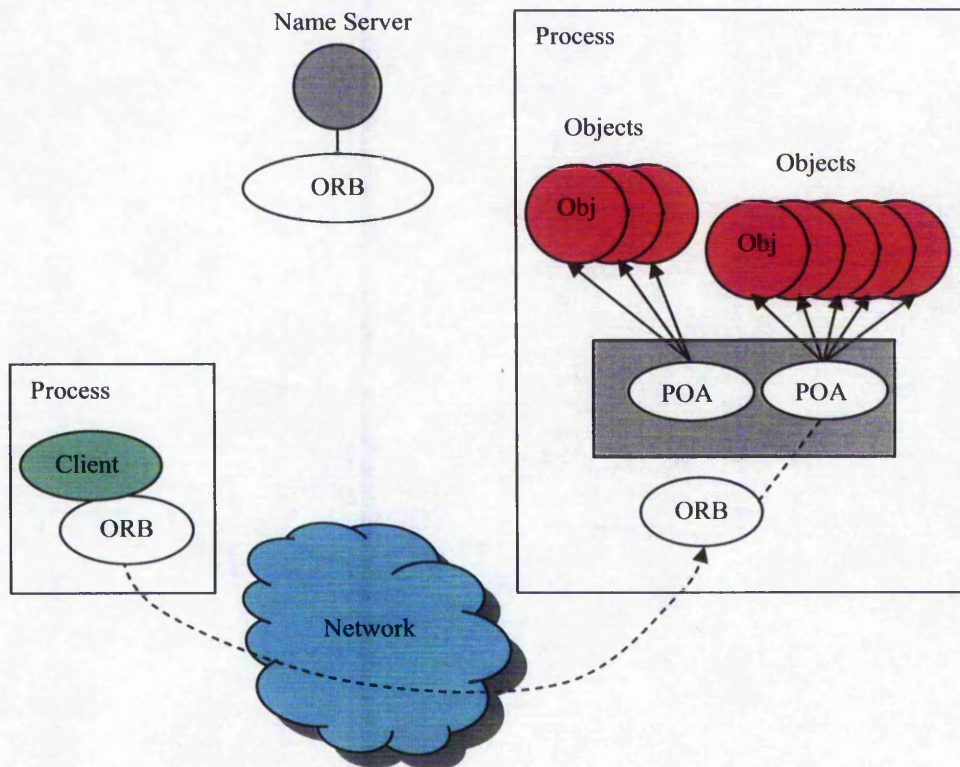


Figure 4.8 A POA's role in Client-Object communication

4.3 Distributed System Framework of the Gear Design Optimisation

4.3.1 Basic Requirements

To develop the Web-based system for gear design optimisation, it is necessary to consider the following basic requirements:

1. A designer wants to conduct gear design to find the rational tuning parameters in consideration of multiple aspects in design and manufacture, within limited time without augmenting his own system. Thus the designer might use remote design resources to enhance his own design ability.
2. The program providers who have design and analysis services available might prefer their own programs to be only invoked and executed remotely rather than downloaded. The analysis codes should reside with the experts, i.e. providers, and the analysis program should execute on providers' computers.

3. The program may be developed in diverse program languages and run on diverse platforms or in an old environment but needs to be used continually.
4. The analysis experts possess a new version of analysis software and hope others will use it.
5. Designers need to invoke remote resources dynamically according to the design objectives.

4.3.2 Architecture of the Distributed System

In order to meet the above-mentioned requirements, a layered approach is adopted in the framework, where there are four layers, namely, the interface layer, the application layer, the data layer, and the Web server, as shown in Figure 4.9.

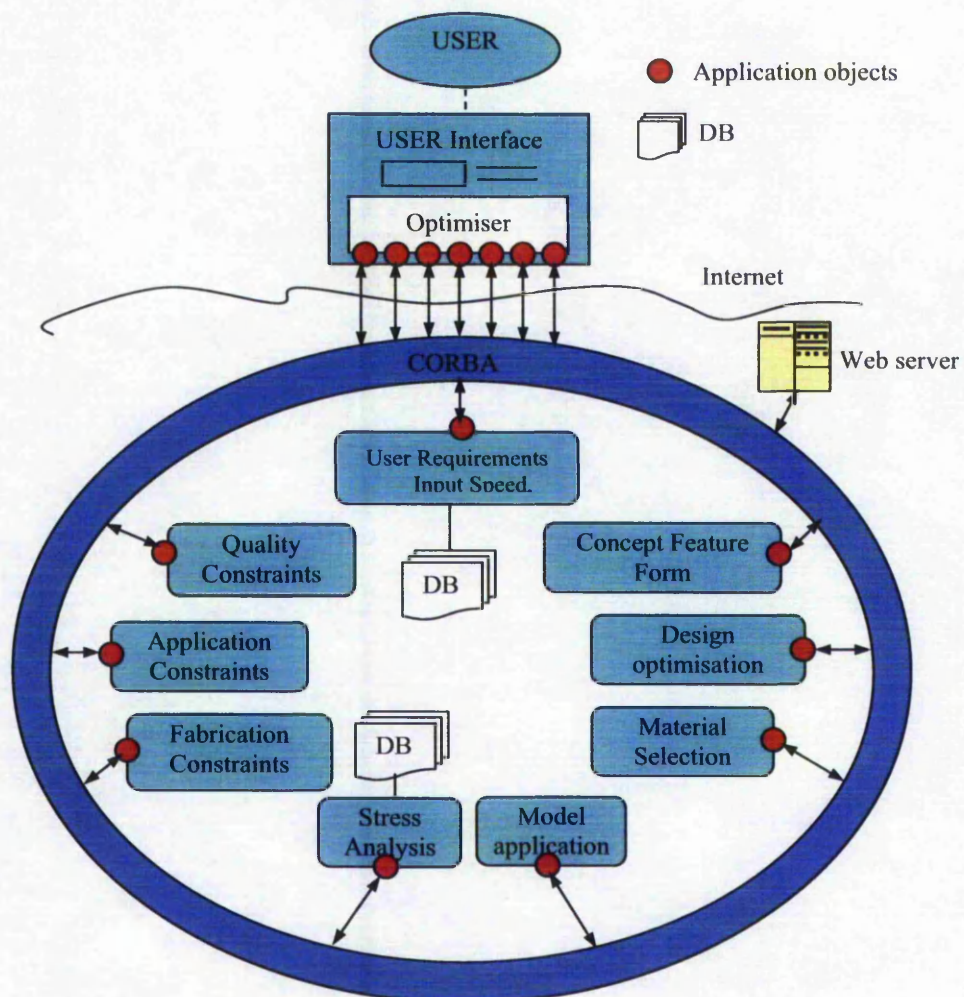


Figure 4.9 Architecture of distributed collaborative design optimisation

The front layer is the interface layer, a unified graphical user interface. Using the interface, users can interact with the design environment in a natural way. A design optimiser is included in the user interface to invoke remote applications.

The application layer is a distributed computing environment (DCE) based on CORBA. Programs used for computing, modelling, analysis, manufacture, and so on are encapsulated as application objects and linked to the CORBA communication bus, and then can be recognised and invoked within the GUI and optimiser program.

The data layer forms a data warehouse making up of database (DB), knowledge base (KB) and corresponding management programs. It provides data information, status information and control information of the system, including geometry data, and engineering data produced in the evolving process. The technique standards, criterion and generic data used in the whole design period are stored in the knowledge base. The local data could be retrieved by the relative application while global data lies where it can be accessed with each application.

The Web server provides the Web services such as member registration, resources management, and so on.

In the framework, a distributed environment is built upon the Web infrastructure. To provide a generic support for all different applications executed under different platforms and designed by different application programmers, cross-platform CORBA standard is employed as the underlying unifying implementation framework. CORBA supports seamless integration of modules, which are written in different languages and built upon different platforms. In the CORBA-based framework, the client-side application in the front layer may invoke the server-side objects that stand for gear design resources in the application layer. The client application, used as user interface, is an application on Windows while design applications may be ported on different platforms such as Linux, Windows or others. Through the CORBA communication bus the User Interface program cannot only directly invoke heterogeneous

applications in a point-to-point model, but also obtain general management information through a Web server.

Actually, the user interface program communicates with remote applications through an optimiser mechanism included in it. The optimiser is designed and developed to help the user (i.e. designer) to obtain optimal design.

4.3.3 Remote Invocation Mechanism

The graphical user interface is designed as a CORBA client program and linked to all the essential gear design applications or service objects. CORBA IIOP and ORB makes it possible for the interface program to invoke remote application service objects, across all the boundaries in geographical, platforms, operating systems, and programming language aspects. The remote invocation based on the distributed system is implemented as shown in Figure 4.10.

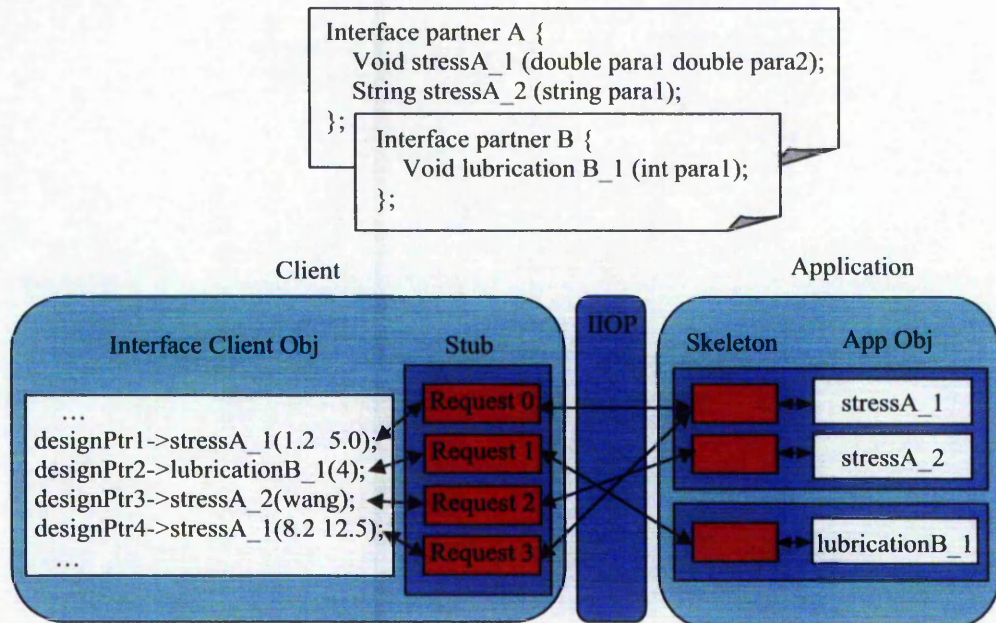


Figure 4.10 Remote invocations based on the distributed system

The interface definitions for gear application service objects, *stressA_1* for gear contact stress calculation, *stressA_2* for gear bending stress calculation, and *lubricationB_1* for lubricating feature calculation, are implemented in IDL. The

interface definition program contains only the input and output parameters, operation name of an object other than the essential code in details. What a client program developer needs to know about the remote object is only what the object is for, reference name, and input and output parameters. The invoking requests for the remote objects are sent in the graphical user interface program through client side ORB (stub) to server side ORB (skeleton) that bounds to the objects. The objects execute on their original environment and the results are retrieved by the client procedure. The system facilitates many forms of parallel mechanism. The system allows multiple users from different sites to simultaneously invoke the same objects.

4.4 Working Procedure of Distributed Gear Design Optimisation

The gear design is abstracted into a design optimisation set, namely design variables, design objectives, and design constraints. The design engineers must make the design services available over the Internet through the CORBA-ware approaches. The design variables are improved by the design optimisation procedure including multiple criteria evaluation and constraints violation inspection from multiple concerns. During the design evolving process, the distributed gear design optimisation (DGDO) system might invoke local and remote calculation programs for the analysis and evaluation of the every candidate design solution.

The variable dimension mechanism of the optimisation algorithm in this system allows a designer to select any variable set among up to 9 design variables. Multiple objective optimisation structure, as a decision maker aid, helps the designer to find multiple feasible solutions within a broad region of solution space. Every objective is linked to the local or remote calculation routine. According to the design objectives, the decision maker aid, included in the client program i.e. GUI, will automatically call the corresponding calculation operation locally or remotely. In addition to normal output data, a CAD data file parameterised by the improved variables can be automatically generated for visualisation and further utilisation.

The working procedure of the DGDO system is shown in Figure 4.11. GA is used as the evolution algorithm, and multi-disciplinary service tools are used as the evaluation mechanism. The model runs recursively till the best designs are identified. All possible design evolutions can be visualised and their performance can be predicted. All the activities are functionally divided into four groups: parameter input and identification, design evolution, performance evaluation and constraint penalty, and design view.

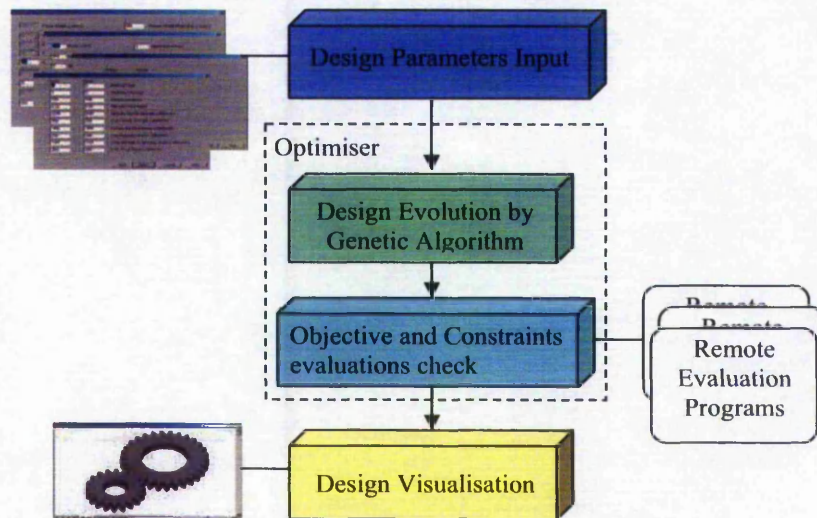


Figure 4.11 Working procedure of DGDO

4.4.1 Design Parameters Input

To start a gear design process, a designer needs to input design parameters to specify design objectives, the design variables, and basic design requirements, in the corresponding data input dialogues. To help a user to fully and exactly understand the meaning of so many parameters in these data input dialogues, an on-line help function is facilitated in the system so that the definition and specifications about every parameter can be found in the pop-up help dialogue by pressing the F1 button.

4.4.2 Design Evolution by Genetic Algorithm

After finishing the parameters input and product data preparation, the system proceeds to the design evolution stage. The optimiser is then activated and carries out the design evolution. This application relates to helical gears, of which spur gears may be considered a special case. Besides computing basic spur and helical geometrical

parameters, tooth cutting tool tip design in the manufacturing process and complicated design on addendum modification for spur and helical gears have been included in this application. The optimiser is implemented in C++, which is ported on the Windows system.

Since this is a mixed continuous – discrete variable problem, GA is used to explore the design space. A set of variables is represented as an individual (chromosome), coded in binary in the GA program. The first generation of individuals (chromosome) is generated randomly over the whole search space. The evolution runs recursively through three evolutionary operations, e.g. selection, crossover, and mutation, till the best designs are identified. The details about GA-based optimisation technique and the results analysis are presented in Chapter 5.

4.4.3 Objective and Constraints Evaluation

Every individual in every generation is evaluated with the fitness function. The fitness calculation involves not only objectives but also a penalty function related to the constraints. The gear design evolution model is designed to address the synthesis design from multiple disciplines. The objectives and constraints are calculated according to the formulas, knowledge, standard and experience in the domain. There are up to six design objectives and twenty-four inequality constraints. All the relevant calculations are illustrated in detail in Section 5.1 and in Appendix C. The design optimisation procedure invokes the objective and constraint evaluation programs to calculate the fitness function to evaluate every set of solutions, in terms of the CORBA communications mechanism, as shown in Figure 4.12.

In the optimisation process, the optimiser invokes the remote evaluation service procedures. This kind of communication in the CORBA-based distributed system is transparent, just as a local function invocation, no matter in which language, it is written, or running in which platforms. The stress analysis procedure and manufacture constraint procedure are ported in Linux and written in C/C++. The other quality calculation procedures are located on another Windows system and written in Java.

The implementation of the communication mechanism provided in the system is described in Section 5.2.3.

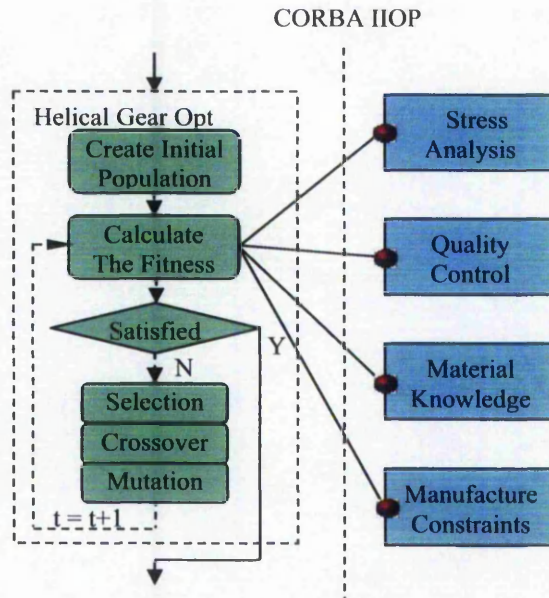


Figure 4.12 Optimisation procedure and remote evaluation programs invocation

4.4.4 Design Visualisation

The system facilitates a 2-D model generation service tool to assist users in the design process by visualising the best design. Once the designer changes design configurations and parameters, the system has the ability to rapidly recalculate and redesign the gear system to reach the desired design configuration.

The automatic gear tooth generation tool is developed to establish an intelligent design environment. The tool, written in C++, covers a complete design procedure that exports DXF files to a DXF-aware CAD tools such as AutoCAD, or Free Viewer, as shown in Figure 4.13. The 2-D gear tooth profile is calculated using the formulas provided in Dudley [102].

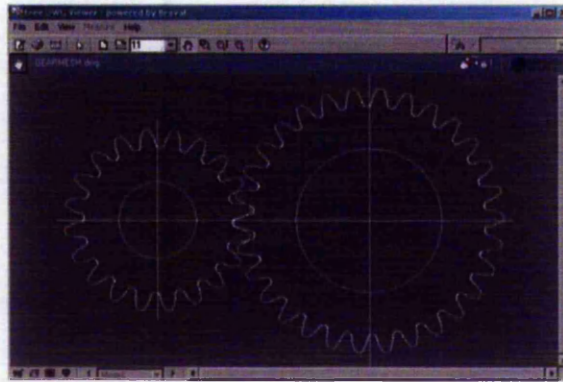


Figure 4.13 An example of an automatically generated 2-D gear profile

Based on the generated 2-D DXF file the 3-D modelling and structural design can be produced using most commercial CAD software, such as AutoCAD or Pro-E. The 3-D solid model of gears is shown in Figure 4.14.

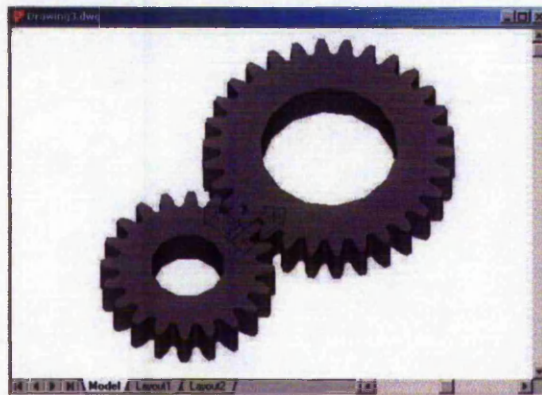


Figure 4.14 A 3-D solid model of gears

4.5 Automatic Gear Tooth Generation

4.5.1 Gear Profile Calculation Formula

In this application it is necessary to generate the exact 2-D gear tooth profile. The gear profile can be split into three distinct regions, as shown in Figure 4.15. The first part of the gear tooth profile is the working portion (involute), cut by the linear flank of the rack cutter. The second part is the root circle, generated by the tip of the rack. The third portion, the trochoidal fillet, connects the working portion and the root circle.

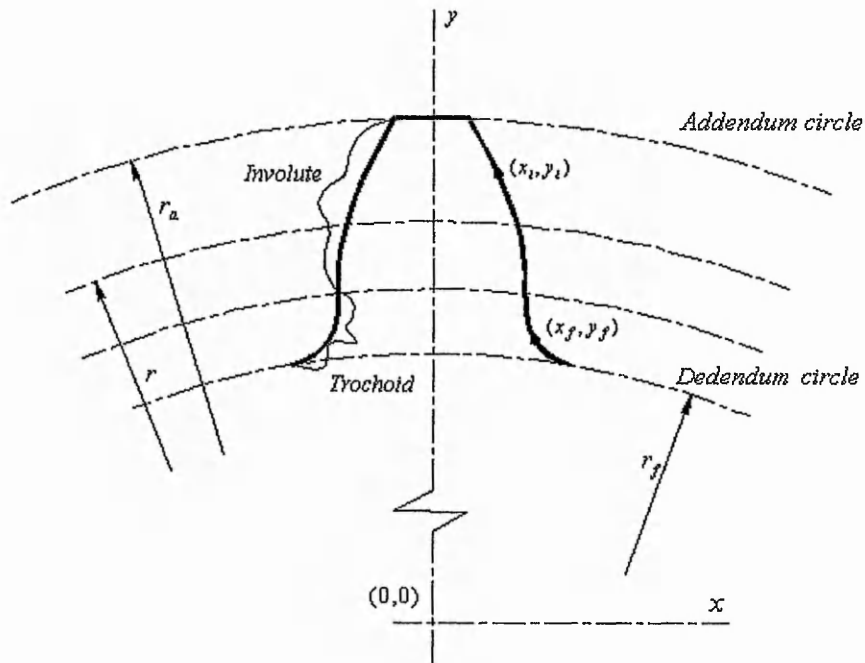


Figure 4.15 Gear tooth profile

The following formulas are used for gear profile calculation, where the symbols have their usual meanings as defined for example in Dudley [102].

1. Involute profile

$$\begin{cases} \theta_i = \tan \alpha_i - \alpha \\ r_i = r_b / \cos \alpha_i \end{cases} \quad (4-1)$$

$$\begin{cases} x_i = r_i \sin \theta_i \\ y_i = r_i \cos \theta_i \end{cases} \quad (4-2)$$

2. Fillet profile

$$\begin{cases} x_f = \left(\frac{\pi - S_{n_0}}{2} + (h_{a_0} - \rho_{a_0}) \tan \alpha + \frac{\rho_{a_0} - \delta_0}{\cos \alpha} \right) / \cos \psi \\ y_f = X_g - (h_{a_0} - \rho_{a_0}) - \rho_{a_0} \sin \alpha \end{cases} \quad (4-3)$$

3. Root circle

$$r_f = r - h_{af} \quad (4-4)$$

4. Tooth tip thickness

$$s_a = d_a \left(\frac{s}{mz} + \text{inv}\alpha - \text{inv}\alpha_a \right) \quad (4-5)$$

4.5.2 DXF Data File Format of the Gear Tooth Profile

The 2-D gear tooth profile generation is designed to export a DXF file. DXF is an abbreviation of Data Exchange File, a two-dimensional graphics file format supported by virtually all PC-based CAD products. AutoDesk created it for the AutoCAD system, but it can be also viewed in other popular commercial CAD software such as UG and Pro-E or even free software such as FreeViewer, or QCad on Linux.

The DXF format is a tagged data representation of all the information contained in an AutoCAD drawing file. Tagged data means that each data element in the file is preceded by an integer number that is called a group code. A group code's value indicates which type of data element follows. This value also indicates the meaning of a data element for a given object (or record) type. Virtually all user-specified information in a drawing file can be represented in DXF format. The profile curve is represented by numbers of *polyline* entity.

4.6 Implementation of GUI on Windows

In modern software development, an algorithm program and its service GUI are often separately developed by different developers, and even on different sites, on different platforms and different languages. In this project, the GUI is developed in VC++ and running on Microsoft Windows. The GUI provides a unified interface for users to conduct design virtually. The GUI also includes an optimiser routine, which is a designed CORBA client invoking remote CORBA objects, i.e. evaluation programs. The user does not need to know any technical details about CORBA-based remote communication.

4.6.1 Design Objectives Identification

There are six design objectives, in this gear optimisation system, to be selected. A user can identify the design objectives and relevant weighting coefficients by using the

interface, as shown in Figure 4.16. Each of the objectives is associated with a local or remote evaluation program.

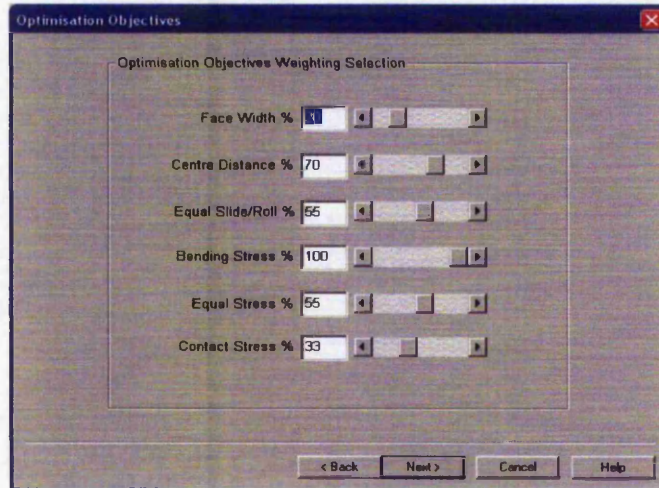


Figure 4.16 Design objectives identification

4.6.2 Design Variables Configuration

There are in total up to 9 optimisation variables defined in this gear design application. In order to provide the user with more flexible design spaces, the system supports the dynamical design for less than 9, or up to 9, of the variables, e.g. dimension variation. In the case of fixed centre distance, the variables z_1 and x_2 are not independent, and are determined from other variables in the calculation program.

For different design circumstances, a user may select different design variables from these nine variables. The selected variables will be included in the optimisation process, therefore, the user does not need to input values for them in the Geometry Parameters data input dialogue. However for any unselected variables, a user must provide proper constant values for them in the data input dialogue. In the actual design example shown in Figure 4.17, Face Width, Module and Addendum Coefficient are three selected variables, so their input columns are disabled, while the other six variables are unselected variables, so their input columns are enabled.

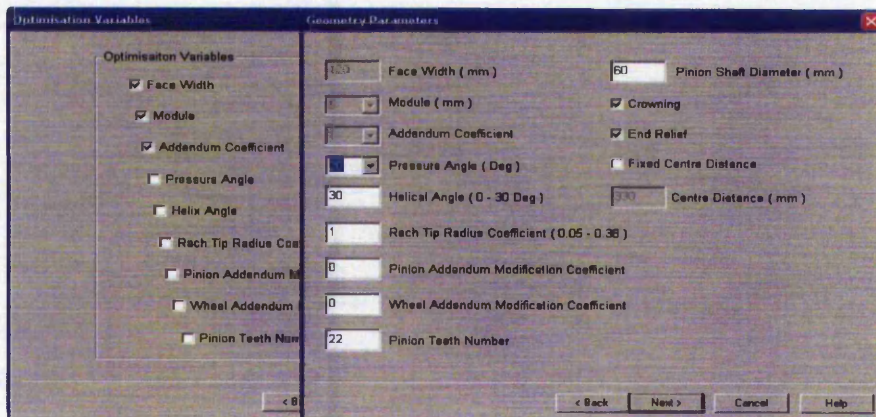


Figure 4.17 Selected and unselected optimisation variables

4.6.3 Specify Other Design Requirements

In addition to the design objectives and design variables, the user still needs to specify other basic design requirements, including application parameters, quality parameters and material parameters. Application parameters include Power, Input Speed, Gear Ratio, etc, in total 14 parameters. Quality parameters include Material Quality, Manufacturing Accuracy, Flank Surface Roughness and Root Surface Roughness for both pinion and wheel gears. Material parameters include Material Type, Hardness Process, Surface Hardness, etc, in total 10 parameters for both pinion and wheel gears. Application, quality and material parameters data input dialogue is shown in Figure 4.18.

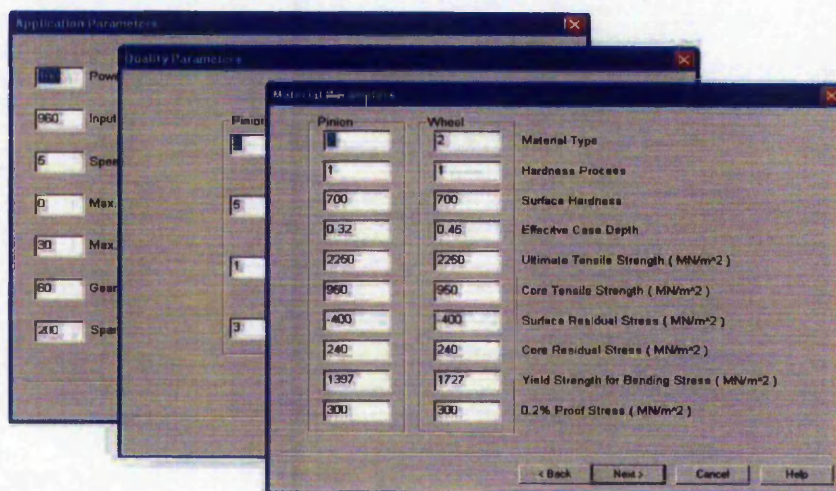


Figure 4.18 Application, quality and material parameters data input

4.6.4 On-line Help

To help users to fully understand every menu item in each dialogue, and input so many parameters correctly, an on-line help is provided. During the input process, pressing the F1 button, or the Help button on the bottom of each dialogue, can activate the on-line help to see the relative knowledge about the current dialogue and the knowledge about these parameters, as shown in Figure 4.19.

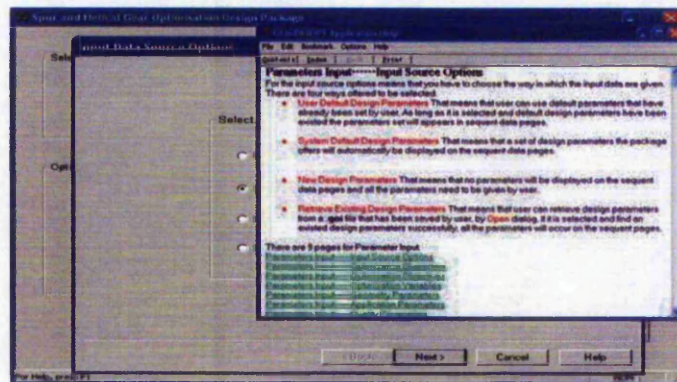


Figure 4.19 Pop-up on-line help

4.6.5 Input Data Selection

As mentioned above, there so many design parameters, to be given in certain values or in certain options, required for user to input, that it is not a straightforward task to input all these parameters. To facilitate a user to input these parameters correctly and quickly, the application program provides four flexible data input selections, include User Default Design Parameters, System Default Design Parameters, New Design Parameters, and Retrieving Existing Design Parameters, as shown in Figure 4.20.

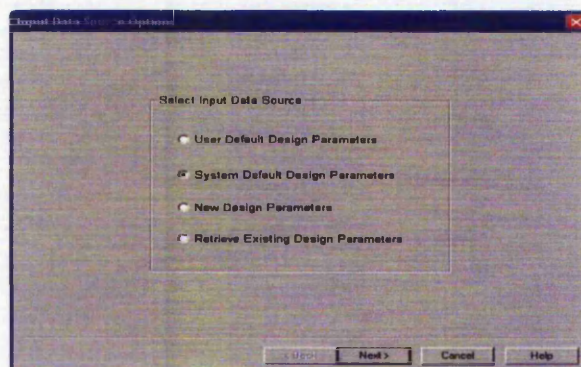


Figure 4.20 Input data selection

If the “Retrieving Existing Design Parameters” option is chosen, a File Open system pop up dialogue will appear for the user to retrieve these design parameters from an existing file saved before, as shown in Figure 4.21.

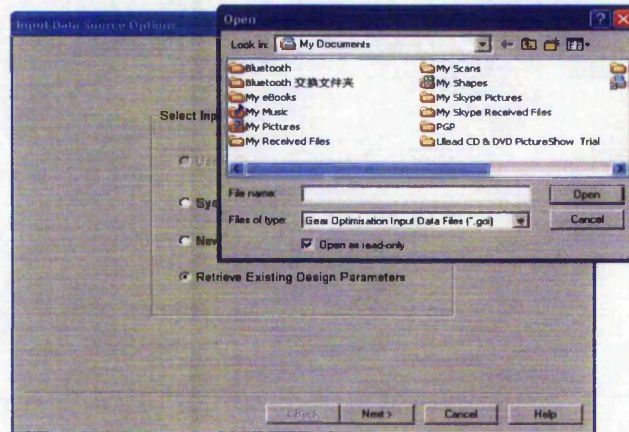


Figure 4.21 Retrieve Existing Design Parameters file open dialog

4.6.6 Calling Optimiser Program

The Optimiser, i.e. optimisation algorithm program, is implemented separately from the GUI procedure. Pressing the “Optimisation Start” button, as shown in Figure 4.22, will cause the algorithm program to be invoked. The optimisation algorithm execution is encapsulated within the “Process” and managed in the GUI program running on a Windows application environment.

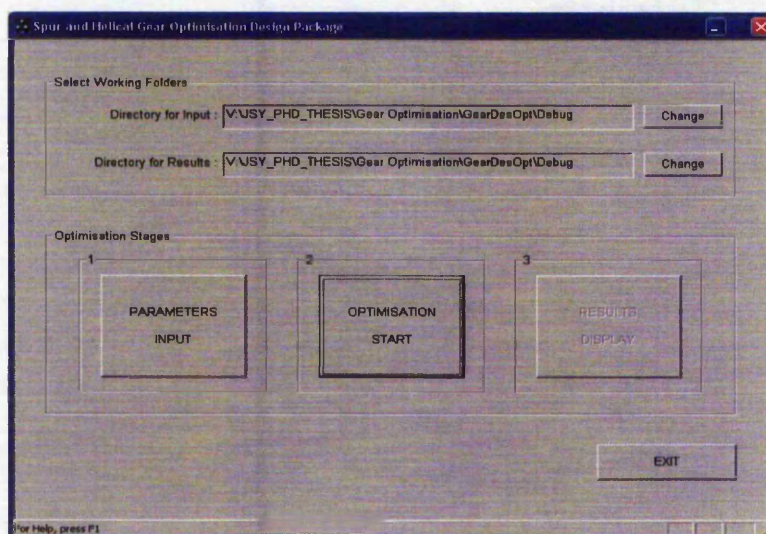


Figure 4.22 Calling algorithm program

Within the optimising process, the optimiser program is needed to invoke evaluation programs that might be local or remote. In this distributed system, the optimiser is designed as a CORBA client while all the evaluation programs are encapsulated as CORBA objects. Therefore the optimiser program can communicate with all the evaluation programs in a client-server model, based on a CORBA communication mechanism, as required. The communication mechanism is illustrated in Section 4.3.3 while the development issues about remote evaluation object implementation and the optimiser client are described in the following sections.

4.7 Implementation of Evaluation Programs as CORBA Object on Linux

In this project, the optimiser application, which is used as decision maker aid, is developed separately from the GUI program. Both of them are ported on the same Microsoft Windows machine. During the process of running the algorithm, the evaluation functions for calculating fitness of GA will be invoked a large number of times. These programs or procedures might be located on different platforms at other sites. In order to invoke these programs ported on different machines over the Internet, each evaluation program needs to be wrapped as a CORBA object. Technical details about the wrapping procedure of turning an evaluation program into a CORBA object on Linux are presented in this section while the implementation of the client process on Windows is described in Section 4.8.

The general procedure for developing a distributed CORBA application is described in Chapter 4. In this application paradigm, 'OmniORB' is used as a tool for developing a CORBA client on Windows and evaluation object program on Linux (RedHat Linux 9.0). Calculations on the specific sliding ratio of gears, related to the anti-wearing properties, are implemented on Linux. These are written in C/C++ and debugged by GCC and Qt developing environment.

'OmniORB' is an Object Request Broker (ORB), which implements the Common Object Request Broker Architecture (CORBA). It is freely available under the terms of the GNU free project on Linux. It is one of only three ORBs to have been awarded the

Open Group's Open Brand for CORBA. It supports the C++ and Python language bindings, is fully multithreaded, uses IIOP as the native transport, and comes complete with a COS Naming Service. OmniORB is possibly the fastest available C++ ORB. This is the main reason that it is chosen in this project.

OmniORB requires *Packages Python* for building the developing environment because OmniORB is downloaded on Linux in the form of source code. After running a proper configuration file especially for the Linux platform, the source code of OmniORB is built using GCC, a compiler of C++ on Linux, and Python, a kind of programming language. The makefiles in the OmniORB package is executed so that the installation is implemented, with the setting up for the lib path and bin path.

Configuring the Naming service, a facility of CORBA services, is another important part of setting up for the developing and deploying of CORBA objects. 'OmniNames' is the command of starting the Omni ORB Naming service. The file omniORB.cfg, is produced for recording all the information of the Naming service. An environment variable is set for pointing to the location of the omniORB.cfg.

Just as all the other CORBA-based applications, interface definition for the object is first written. It contains the input and output data, and name of an object, i.e. an operation. 'Omniidl' is an IDL compiler for creating ORB files: Functionname.hh and Functionname.cc. Implementation codes of the object need to be written and it contains the essential code of a legacy program.

Compiling and linking for all the ORB files and the object implementation files could be implemented by a makefile that is predefined by developers themselves.

4.8 Implementation of Algorithm Application as CORBA Client on Windows

On the client side of Windows, OmniORB can be downloaded in the Win32 binary distribution and installed without building. The remaining work in the developing environment is only the Naming service setting up. Running the tool REGEDT.EXE

for opening Windows “Registry Editor” is to set the location of the Naming service. In this application, the value of the registration key for OmniORB is set in the machine number hosting the remote object, e.g. Linux machine’s IP address number as shown in Figure 4.23.

As a CORBA client procedure, it should contain the initiation of ORB and binding of remote object. The remote Naming service on the Linux platform is started and the object is running and getting ready for invocation while the client program could invoke the remote object by using the registration value for the remote Name service, including the referencing information of the object, which will further point to the object.

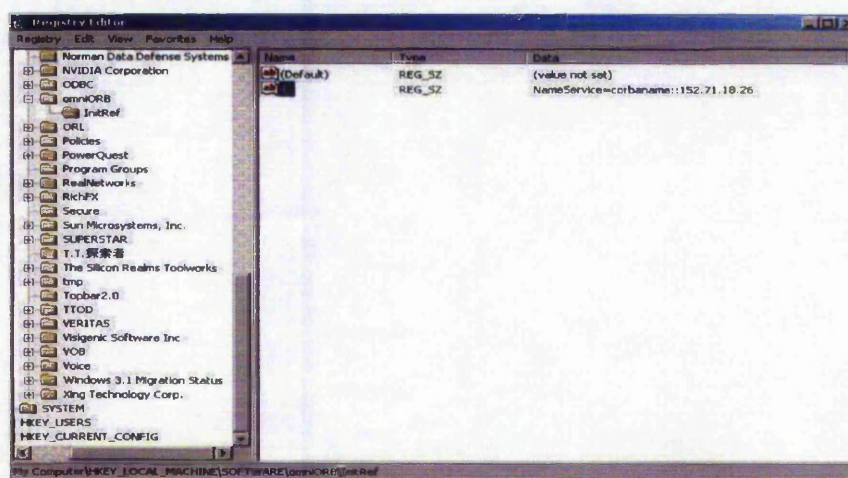


Figure 4.23 Registration of remote naming server

In addition to the basic setting up of the CORBA Naming service, another key issue is how to combine the CORBA client procedure with the legacy code. There are three ways to do this.

1. A CORBA client procedure is developed separately from the algorithm program. The execution file of the client procedure is inserted into the algorithm program in the form of Windows “Process”, as shown in Figure 4.24.

In this way, the CORBA client development could be separated from the development of an algorithm. However starting a new process could affect the execution speed of the entire application procedure.

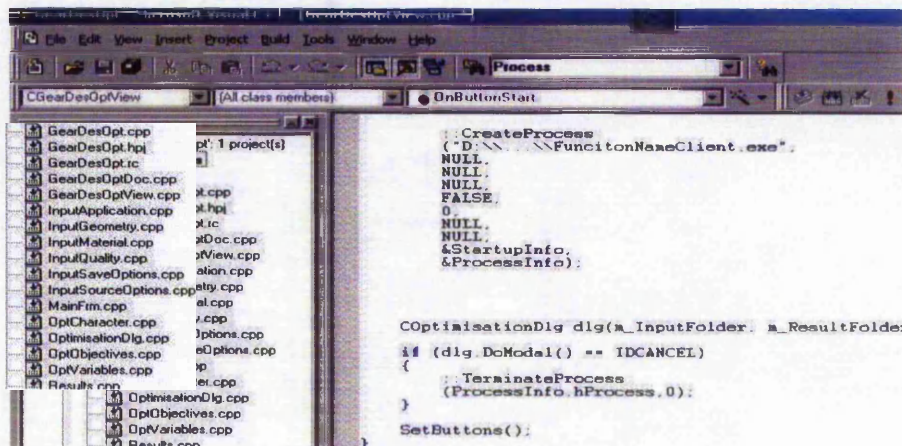


Figure 4.24 Client procedure is added in a “Process”

2. The algorithm procedure is inserted into a CORBA client program and debugged in command line through CORBA-ware makefiles.
3. A CORBA client program is inserted into an algorithm application and debugged in the VC++ environment with the project setting up for the CORBA special procedure.

In both of the latter ways, a new process is avoided which means better execution speed. Which form is chosen depends on the developer’s expertise and preferences and the application features.

4.9 Summary

This chapter presents a distributed system based on CORBA for multiple objective design optimisations of gears. The system consists of four layers: user interface layer, application layer, data layer, and the Web server layer. All the program resources are encapsulated into CORBA-ware distributed components i.e. objects, which are located on the application layer. The user interface layer is a unified user interface, designed as a CORBA client, through which any remote objects can be invoked. The Web server is used for providing resource administrative services. The data layer contains backend data resources, which are linked to the design applications, i.e. CORBA objects, or directly linked to the client program through the CORBA communication bus.

A user of the system merely interacts with the unified interface to conduct complex gear design and does not need to know any technical details about communications between the interface and remote resources.

In order to conduct complex gear design optimisation, the user needs to collaborate with the collocated models or programs that represent the different domain interests. A GA-based design optimisation mechanism is facilitated to help a user to leverage the different interests to obtain rational solutions. The optimisation mechanism is designed as a CORBA client procedure, through which the interface may invoke multiple design resources, located at different sites, ported on different computer platforms (such as Windows and Linux) and written in different languages (C/C++ and Java).

Through the integrated design environment a user (engineering designer) can conduct all the design activities: inputting design parameters activating optimiser (i.e. design maker) that further invokes remote resources, and viewing design results both in text data and graphics.

In this chapter, the developing techniques about the communication implementation between Linux and Windows are also presented, including environment setting up, communication approach, combination methods between CORBA-related procedure and legacy codes, etc.

Chapter 5 Gear Design Optimisation Using Genetic Algorithms

5.1 Introduction

In the distributed gear design optimisation system presented in Chapter 4, a design optimisation mechanism, i.e. an optimiser is designed to help the user to conduct the design optimisation, as a design aid. In this chapter, the design optimisation approaches and calculations are presented in detail.

In the real world, product design is commonly subjected to many different interests. In gear design, examples of interests are less space size, lower bending and contact stresses, high performance for resistance to friction, as described in Appendix C. In an ideal case, geometrical parameter specification of gears should involve considerations of all engineers' interests; however, this is impossible because some of these interests conflict with each other. Consequently, the design decision must reach a compromise between the different considerations. Designers within the field of gear design are thus facing a complex decision problem involving tremendous calculations of objective evaluations and conflicting interests based on economic, structural, manufacturing or application criteria.

The main goal of the work in this chapter is the implementation and validation of a design optimisation mechanism, which can be used to support spur and helical gear design optimisation. More especially the following work will be done:

1. Gear design optimisation problem is modelled.
2. Genetic algorithms are identified and investigated.
3. Multiple objective optimisations are implemented.
4. Simulated annealing is employed to construct a variable penalty function.
5. Resultant analysis of gear design based on the above techniques.

5.2 Gear Design Optimisation Model

Gear design is one of the classical topics of mechanical engineering design. The classical route followed for the design of gears is to appeal to standards, such as BS, AGMA, DIN or ISO [92-98]. These standards are based on extremely large collections of results and empirical rules from practical experience in a vast range of engineering applications. They provide a set of formulae, rules and charts to design the gearing taking into account various working conditions and several aspects of their performance, such as the power level, noise, lubrication conditions, wear rate, likelihood of impact, pitting, and corrosion. Nevertheless, actual gear design involves very difficult and complex calculations and trial and error, and thus often requires an iterative process to determine those design parameters that would satisfy performance and strength requirements, which would result in an efficient and reliable operation for gear transmission system.

In this project, a gear design problem is used as case study to explore the possibility of the proposed collaborative system for improving design efficiency. In this section, relative formulae, and experience in gear performance, gear strength, and wear resistance used in this research are presented. Gear geometrical design with rack shift (addendum modification) and its relevant considerations are identified. The calculations use procedures, algorithms and data from standards ANSI, ISO, DIN, BS and specialised literature.

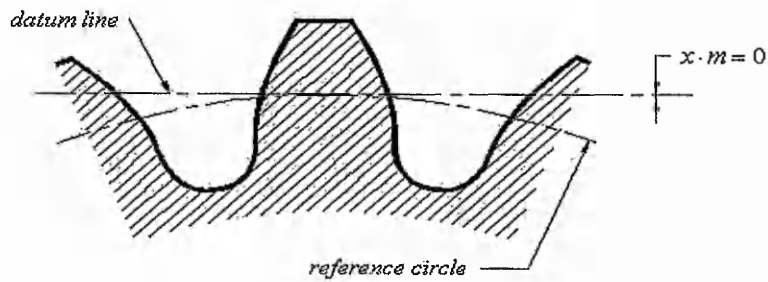
This application relates more specifically to helical gears, of which spur gears may be considered as a special case. In addition to the basic design, a complicated design addendum modification of spur and helical gears is also included in this application.

5.2.1 Addendum Modification Design

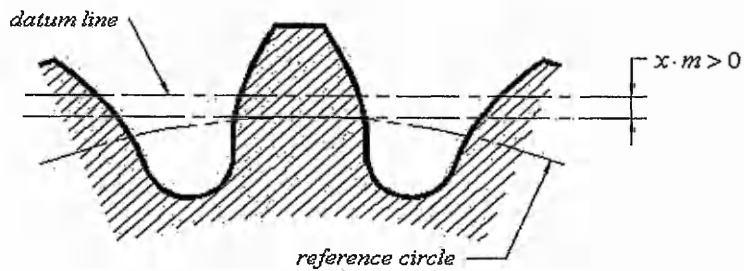
5.2.1.1 Addendum Modification

When gears are produced by a typical generating process, the datum line of the basic rack need not necessarily form a tangent to the reference circle. The tooth profile can be formed by shifting the datum line from the tangential position. The radial

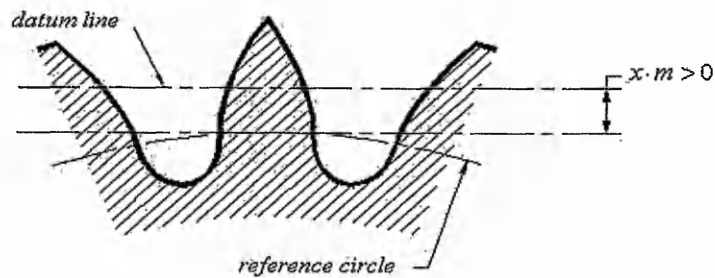
displacement from the tangential position is termed addendum modification, i.e. rack shift. Four different instances of addendum modification are shown in Figure 5.1.



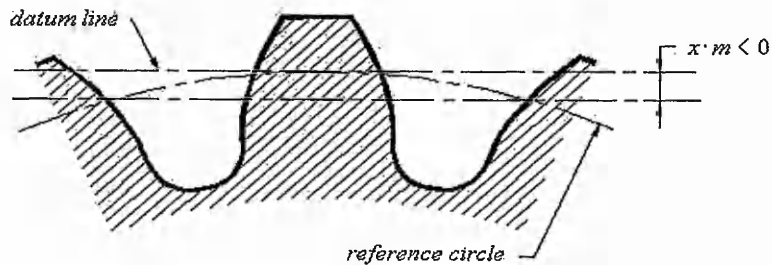
5.1a No addendum modification



5.1b Favourable positive addendum modification



5.1c Excessive positive addendum modification



5.1d Negative addendum modification

Figure 5.1 Four instances of addendum modification

The displacement is considered positive when in the direction away from the centre of the gear and negative when nearer towards the centre. Regardless of positive or negative displacement, the involute shape of the tooth profile is retained. The tooth profile, however, may use a different form of involute curve in different cases of displacement. For positive displacement, the part further from the origin of the same involute is used, while for the negative displacement, the part nearer to the origin of the same involute is used.

The load carrying capacity of the teeth without addendum modification shown in Figure 5.1a can be improved by the positive addendum modification shown in Figure 5.1b. However, an extremely large addendum modification results in an unsuitable tooth form with pointed teeth, as shown in Figure 5.1c. A negative addendum modification is shown in Figure 5.1d.

The addendum modification coefficient x is the amount of the addendum modification divided by the module. Thus, the amount of the addendum modification is equal to $x * m$.

5.2.1.2 Effect of Addendum Modification on the Tooth Form and its Application

Firstly, the effect of addendum modification on the tooth form is particularly significant for its load carrying capacity. The load carrying capacity of gears with appropriate addendum modification could be improved by over 20% with respect to a standard gear, with no need for special machine tooling, cutting tools or process technology. The following characteristics are the factors of the effect of addendum modification on load carrying capacity.

- a) The profile angle α' , because of the relationship between the mean radius of curvature of the tooth flanks and the contact load capacity. If $x \cdot m > 0$ then the profile angle α' is greater than pressure angle α on the reference circle of standard gear.
- b) The tooth root thickness, because of the relationship between the modulus of section and bending strength at the root of the tooth.

- c) The fillet radius at the critical point for bending, as at this point a rapid change in cross-section results in stress concentration.
- d) The crest width, as excessive shear stress at the tip is undesirable, particularly in surface hardened gears.

In addition to the effect on the load carrying capacity, the addendum of modification is also utilised in the following application circumstances:

- a) To avoid the cutter interference at the root of gear. In the case of $z < z_{\min}$, correct positive addendum modification avoids cutter interference.
- b) To fit a given centre distance. In the case of z_1 and z_2 given, it is possible to adapt to the given centre distance by adjusting the addendum modification coefficients x_1, x_2 .
- c) To repair the worn gear pair. In gear power transmission, if both the pinion (smaller one) and the wheel (bigger one) have been worn, the common solution to this problem is to repair the wheel and replace a new pinion. The new pinion with positive addendum modification can mate with the repaired wheel with the renewed profile of negative addendum modification. In this way, the manufacturing cost for a new wheel can be saved.
- d) Improve the performance in other aspects. The transmission with positive addendum modification leads to the improvement of contact and bending strength. Choosing appropriate addendum coefficients could reduce the specific sliding ratio, and the lower specific sliding ratio could be beneficial to the micropitting resistance.

5.2.1.3 Consideration of Determining the Addendum Modification Coefficients

Determination of gear addendum modification coefficients is a complex process. It affects geometric, kinematic and strength characteristics as well. Many practical graphs and tables for fulfilling some functional requirements have been applied to conduct the modification coefficient design [101, 102]. However most of these address only some aspects of consideration, such as a requirement of high resistance against

wear, requirement for a high contact loading capacity, protection from undercutting of the teeth, or protection from tapering of the teeth.

It is difficult to determine the addendum modification coefficients simultaneously considering functional requirements and limiting conditions. The common way to do this is by firstly making the initial design option according to an experienced table, graphs or formula, and then, conducting analysis of strength and stress, repeating re-option and then re-analysis until receiving the satisfied results.

In this project, determination for the addendum modification coefficients and other parameters is implemented by the design optimisation to satisfy multiple functional objectives under multiple constraint conditions to determine if it is possible to obtain a quick and precise design.

Possible relative functional requirements for choosing the addendum modification coefficients are

- a) Minimising the bending stress and the contact stress for high load capacity
- b) Minimising the difference between the maximum of specific sliding ratio at both the pinion and wheel for high resistance against wear

Under the following constraint conditions:

- a) Preventing undercutting of teeth during manufacturing
- b) Preventing interference of gearing.
- c) Checking the minimum of tooth thickness on the tip diameter
- d) Checking the minimum of total contact ratio

5.2.2 Gear Design Optimisation Model Considering Addendum Modification

The gear design optimisation model consists of six objectives, nine design variables, and twenty-four inequality constraints. The formulas in this section have been taken from *MAAG Gear Design* [101], *Dudley's Gear Handbook* [102] and *British Standard*

436 [98]. More details about the gear knowledge and calculations are given in Appendix C.

5.2.2.1 Design variables

Up to 9 of the variables for the design of spur and helical gears are taken into consideration:

- *Face width coefficient* ϕ_d : The ratio of face width to pitch diameter of the pinion
- *Module* m : Standard value list
- *Addendum coefficient* h_{ap} : Standard value list
- *Pressure angle* α : Standard value list
- *Helical angle* β
- *Rack Tip Radius coefficient* ρ_{FP} : the product of ρ_{FP} and m is the rack tip radius
- *Pinion Addendum Coefficient* x_1
- *Wheel Addendum Coefficient* x_2
- *Pinion Tooth Number* z_1

In the case of fixed centre distance, the *Wheel Addendum Coefficient* x_2 and *Pinion Tooth Number* z_1 are not independent variables any more. x_2 is determined by the *Pinion Addendum Coefficient* x_1 and the fixed centre distance. z_1 is determined by the *Module* m and the fixed centre distance.

5.2.2.2 Objectives

The optimisation process adjusts parameters that have effects on the characteristics of the gear to fulfil the following criteria:

1. Minimising face width

$$f_1(\phi_d, m, \beta, z_1) = B = \phi_d \frac{mz_1}{\cos \beta} \quad (5-1)$$

where $f_1(\phi_d, m, \beta, z_1)$ is the face width function.

2. Minimising the centre distance of a gear pair for a variable centre distance. If the centre distance of a gear pair is given, then the following expression is considered as an equality constraint

$$f_2(m, \alpha, \beta, x_1, x_2) = a = \frac{1}{2} \frac{mz_1}{\cos \beta} (1+u) \frac{\cos \alpha_t}{\cos \alpha'_t} \quad (5-2)$$

where $\tan \alpha_t = \frac{\tan \alpha}{\cos \beta}$, $\text{inv} \alpha'_t = \text{inv} \alpha_t + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha$, with u , representing the transmission ratio, being given.

3. Reducing the bending stress of the pinion

$$f_3(m, h_{ap}, \alpha, \beta, \rho_{fp}, x_1, x_2, z_1) = \sigma_{b1} \quad (5-3)$$

4. Reducing the difference in bending stresses between the pinion and wheel teeth

$$f_4(m, h_{ap}, \alpha, \beta, \rho_{fp}, x_1, x_2, z_1) = \Delta \sigma_b = |\sigma_{b1} - \sigma_{b2}| \quad (5-4)$$

5. Reducing the contact stress

$$f_5(m, h_{ap}, \alpha, \beta, \rho_{fp}, x_1, x_2, z_1) = \sigma_H \quad (5-5)$$

6. Reducing the difference in the tooth tip sliding ratio of the pinion and wheel

$$f_6(m, h_{ap}, \alpha, \beta, \rho_{fp}, x_1, x_2, z_1) = \Delta S_r = |S_{r1} - S_{r2}| \quad (5-6)$$

5.2.2.3 Constraints

The parameters presented in this section, which would be utilised in a conventional gear design approach, and have the same symbols as normally used [14].

1. Constraints on strength

Contact strength

$$g_1 = 1 - \frac{[\sigma_{H1}]}{\sigma_H} \leq 0 \quad (5-7)$$

$$g_2 = 1 - \frac{[\sigma_{H2}]}{\sigma_H} \leq 0 \quad (5-8)$$

Bending strength

$$g_3 = 1 - \frac{[\sigma_{F1}]}{\sigma_{F1}} \leq 0 \quad (5-9)$$

$$g_4 = 1 - \frac{[\sigma_{F2}]}{\sigma_{F2}} \leq 0 \quad (5-10)$$

The permissible contact strength $[\sigma_H]$ should be greater than actual contact strength σ_H ; and the permissible bending strength $[\sigma_F]$ should be greater than actual bending strength σ_F .

2. Cutter interference condition

$$g_5 = h_{ap} + \frac{1}{2 \cos \beta} z_1 \sin^2 \alpha_i - x_1 \leq 0 \quad (5-11)$$

$$g_6 = h_{ap} + \frac{1}{2 \cos \beta} z_2 \sin^2 \alpha_i - x_2 \leq 0 \quad (5-12)$$

3. Tooth tip thickness

$$g_7 = 0.3m - S_{a1} \leq 0 \quad (5-13)$$

$$g_8 = 0.3m - S_{a2} \leq 0 \quad (5-14)$$

4. Interference at the roots of mating gear teeth

$$g_9 = g_{mv} - g'_{mv} \leq 0 \quad (5-15)$$

$$g_{10} = g_{a2} - a \cdot \sin \alpha'_i \leq 0 \quad (5-16)$$

$$g_{11} = 1 - K_A \leq 0 \quad (5-17)$$

$$g_{12} = 1 - K_E \leq 0 \quad (5-18)$$

5. Slide/roll ratio for the gear tooth tip

$$g_{13} = 3 - \frac{K_A}{1 - K_A} \leq 0 \quad (5-19)$$

$$g_{14} = 3 - \frac{K_E}{1 - K_E} \leq 0 \quad (5-20)$$

6. Rack tip fillet radius coefficient limit checking

$$g_{15} = \rho_{fp} - \frac{0.25}{1 - \sin \alpha} \leq 0 \quad (5-21)$$

7. Increasing the contact ratio

$$g_{16} = \varepsilon = \varepsilon_\alpha + \varepsilon_\beta > 1.2 \quad (5-22)$$

8. Limitations on each variable

$$g_{17} : 0.2 \leq \phi_d \leq 1.4 \quad (5-23)$$

$$g_{18} : 1 \leq m \leq 50 \quad (5-24)$$

m is obtained from the predefined list of 35 discrete standard values

$$g_{19} : 0.75 \leq h_{ap} \leq 1.25 \quad (5-25)$$

discrete standard values (0.75, 1, 1.25)

$$g_{20} : 17.5 \leq \alpha \leq 22.5 \quad (5-26)$$

discrete standard values (17.5, 20, 22.5, 24)

$$g_{21} : 0 \leq \beta \leq \text{HelixLimit}, \quad (5-27)$$

where the *Helix Limit* is dependent on the force F_t

$$g_{22} : 0.05 \leq \rho_{fp} \leq 0.4 \quad (5-28)$$

$$g_{23} : -1 \leq x_1 \leq 1 \quad (5-29)$$

$$g_{24} : -1 \leq x_2 \leq 1 \quad (5-30)$$

$$g_{25} : 1 \leq z_1 \leq 127 \quad (5-31)$$

5.3 Implementation of Genetic Algorithms Program

Genetic Algorithms (GAs) differ from traditional optimisation algorithms and search procedures in four important respects:

- They work using an encoding of the control variables, e.g. solution set, rather than the variables themselves.
- They search from one population of solutions to another, rather than from individual to individual.
- They use only objective function information, not derivatives or other auxiliary knowledge.
- They use probabilistic, not deterministic, transition rules.

As general optimisation heuristics, GAs can be used to optimise a wide variety of problems. The only requirements are an encoding of the problem, an evaluation function for the encoding and the problem property that neighbouring solutions have similar fitness to guide the search. In gear design optimisation, the generality of GAs allows spatial relations to be taken into account without the requirement of linearity or continuity of the evaluation function.

5.3.1 Basic Concepts of Genetic Algorithms

5.3.1.1 General Procedure of Genetic Algorithms

Genetic Algorithms are stochastic search techniques based on the mechanism of natural selection and natural genetics. The following list shows the general procedure of a genetic algorithm as described by Mitsuo Gen and RunWei Cheng [103].

Procedure: Genetic Algorithms

begin

 initialise $P(t)$;

$t=0$;

while (not termination condition) **do**

 Evaluate fitness of $P(t)$ of each individual;

 Selection operation to $P(t)$;

 Crossover operation to $P(t)$;

 Mutation operation $P(t)$;

$P(t+1)=P(t)$;

end while

end

Differing from conventional search techniques, Genetic algorithms start with an initial set of random solutions called *population*. Each individual in the population is called a *chromosome*, representing a solution to the problem at hand, e.g. geometrical parameters of a gear. A chromosome is a string of symbols; it is usually, but not necessarily, a binary bit string. The chromosomes evolve through successive iterations, called *generations*. During each generation, firstly, the *fitness* of chromosomes are *evaluated*. Then the *selection* operator is used to select the chromosomes for the next generation according to the fitness values. Fitter chromosomes have higher probabilities of being selected. In order to create the next generation, new chromosomes, called *offspring*, are formed by either merging two chromosomes from the current generation using a *crossover* operator, or modifying a chromosome using a *mutation* operator. Some of the parents are rejected and an equal number of offsprings are accepted as the replacement of these parents so as to keep the population size

constant. After several generations the algorithms converge to the best chromosome, which hopefully represents the optimum, or at least suboptimal, solution to the problem.

5.3.1.2 Operations on Chromosome

During each generation of evolution, the following three operators are used to create the new chromosomes for the next generation:

- Selection which equates to survival of the fittest;
- Crossover which represents mating between individuals;
- Mutation which introduces random modifications.

1. Selection operator

Chromosomes are selected from the population to be parents for crossover and mutation. According to Darwin's theory of evolution the best ones survive to create new offspring. The probabilistic method determines the probability of reproduction for each chromosome based upon its fitness in relation to the rest of the population. The better the chromosomes are, the more chances to be selected they have. The selection process is based upon the associated probability of the genomes and a random factor. There are many methods in selecting the best chromosomes, which are roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. The selection method used in this study is roulette wheel selection.

2. Crossover and mutation operators

After implementing the roulette wheel method of selecting the parent chromosomes for the next generation, crossover and mutation operations will be undertaken. The level of crossover and mutation is a set based upon experience with GAs and trial and error. In this study, the region is generally from 70% to 95%. Multiple random crossover points are used in this research during reproduction of the chromosomes, due to the large numbers of the chromosomes and their non-uniform size.

The purpose of the mutation is to increase the search area and prevent local optimisation. The rate of mutation is varied, ranging from 0.0001~0.1 in an attempt to increase the repeatability of the results. Determining the level of mutation is achieved by trial and error.

In addition to the evolutionary operations including selection, crossover and mutation, population and convergence conditions are also the factors controlling the optimisation process. Further details about this are presented in Appendix D.

5.3.2 The Cascaded Genetic Algorithm

The cascade procedure applied in this study is depicted schematically in Figure 5.2. In this approach, the entire solution space has a sub-region, which in turn may have further sub-regions. The sizes and number of sub-regions, referred to as tiers, depends on the final desired accuracy for the solution. The length of the chromosome is set accordingly and remains constant till termination of the algorithm. In the first tier the GA is used to undertake searching over the entire solution space. In subsequent tiers, the GA is utilised for finer searching as the search space is reduced. Each chromosome of constant length thus represents a higher accuracy solution. The solution is obtained when the program has cascaded through the selected number of tiers.

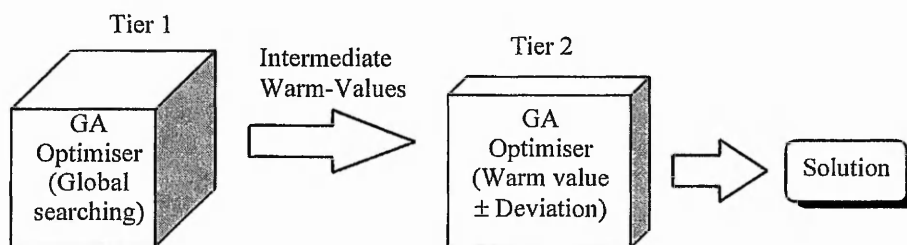


Figure 5.2 Schematic diagram of the proposed Cascaded GA approach

In the cascaded GA, although the chromosome length remains constant, the solution space is redefined and re-initialised at the start of each tier. At the first coarse searching tier, the solution space is defined as the global space. At the finer searching tier, the result from the previous tier is used as the warm value, where upper and lower bounds are obtained through making the warm value plus or minus the deviation value.

If the upper (or lower) bound calculated at the finer tier exceeds (or less than) the upper (or lower) bound given at the first tier, then upper (or lower) bound at the first tier is used as the upper (or lower) bound.

5.3.3 Chromosome Representation

Chromosome is a basic element that Genetic Algorithms deal with. In order to tackle a problem using a GA, candidate solutions must be encoded into chromosome in a suitable form.

The most common way of encoding is a binary string. A binary string represents a chromosome. Each bit in the string can represent some characteristics of the solution. A binary string of a chromosome consists of several sub strings, and each of them is called a gene. A gene represents a variable, and all of these variables together construct a candidate solution.

In addition to the binary methods, there are many other ways of encoding. For example, a chromosome can be encoded directly into integers or float numbers, and sometimes it is useful to encode a chromosome into some permutations. The encoding method mainly depends on the characteristics of problem to be resolved.

5.3.3.1 Chromosome Encoding of Gear Optimisation Application

In this gear optimisation application, the binary chromosome representation has been utilised. A binary string of a chromosome consists of several sub strings, and each of them is called a gene. A gene represents a variable, and all of these variables together construct a candidate solution. There are 9 design variables in this gear optimisation application, so that a chromosome could be encoded like this:

101011 | 101010 | 0011 | 11 | 0100 | 101 | 1001101 | 011100 | 110101
 x_1 x_2 ϕ_d h_{ap} ρ_{fp} α β m z_1

The bit number of each gene depends on the value range of each real variable and the required design accuracy. The variable ranges can be established according to the knowledge of the design problem area. The constraint conditions given by g_{17} to g_{25}

can be used as the initial upper and lower bounds for each variable. In this cascaded framework described in Section 5.3.2, the genes with a constant length are decoded as different real value ranges at each tier (stage). At tier 1, some variables have a larger value range and lower resolution, whereas, at tier 2, smaller value range and high resolution. The encoding and decoding values of the gear design variables in both tiers are shown in the Table 5.1.

Table 5.1 Encoding and decoding values of gear design variables

Variable	Binary bits	Encoding value	Decoding value (tier 1)	Decoding value (tier 2)
x_1	6	0~63	-1~1	$x_{10} \pm 0.25$
x_2	6	0~63	-1~1	$x_{20} \pm 0.25$
ϕ_d	4	0~15	0.2~1.4	$\phi_{d0} \pm 0.3$
h_{ap}	2	0~3	0~3	0~3
ρ_{fp}	4	0~15	0.05~0.4	$\rho_{fp0} \pm 0.09$
α	3	0~7	0~7	0~7
β	7	0~127	0~helixlimit	$\beta_0 \pm 15\%$
m	6	0~63	0~35	$m_0 \pm 7$
z_1	7	0~127	1~127	$z_{10} \pm 7$

These design variables fall into two categories: x_1 , x_2 , ϕ_d , ρ_{fp} and β are continuous, and m , α , h_{ap} and z_1 are discrete. For those continuous variables, the encoding value can be directly mapped into the real value by the following formula:

$$\text{Real value} = \text{Lower limit} + \frac{\text{Upper limit} - \text{Lower limit}}{\text{Maximum encoding value}} \times \text{Encoding value}$$

For those discrete variables, the encoding information refers to a list of pre-defined values from a standard list. The encoding value represents a position in the pre-defined list, and the real value of the variable is retrieved from that position in the pre-defined list.

With regard to the discrete variables m , α and h_{ap} , there are redundant places in the pre-defined list. The gene length of m is 6, hence the maximum places in the pre-

defined list for m is 64 (0-63). The total number of standard value of m is 35 and that means there are 29 redundant places in the list. These redundant places cannot be empty, because for each encoding value, there must be a real value in the corresponding place of the list. To tackle this problem, some more widely used standard values are used to fulfil these redundant places, so that these values have a higher probability of being used. This method is also used for α and h_{ap} .

With regard to the discrete variable z_1 , the real value of z_1 can be calculated by the same formula as the continuous variables. The tooth number z_1 must be an integer, and thus the result of the calculation must be rounded.

5.3.3.2 C++ Bits-field Structure for Chromosome Encoding

In this gear design optimisation program, the number of chromosomes is set by the *population*, which ranges from 100 to 10000, while the *generation*, another variable used to control the process of evolution, sometimes can reach a very large number, e.g. 10000 or even more. During each generation, every chromosome in the population must be the encoded and decoded, which means a substantial amount of computation is involved especially when the *population* and *generation* are in their upper limit. Therefore, the design of the program data structure of a chromosome has a vital effect to the execution efficiency of the program.

Normally, each bit in the binary string is expressed as an integer, and the whole chromosome is expressed as an integer array in the C++ program. So, for the 45-bit chromosome, an integer array with 45 elements is used to express one chromosome. Assume the short integer is used and each integer is two bytes long, the whole chromosome needs 90 bytes of computer memory. When the *population* is big enough, the computer memory used for the chromosome is considerable.

The memory problem is not the only problem of this approach. Another problem of this approach is the computing speed. To decode a variable value from the

chromosome, the program must loop through all those binary bits in a corresponding gene and do several multiplying mathematic calculations.

To overcome the memory and computing speed problems mentioned above, in this study, a built-in C/C++ feature, called a bit field structure, is used to allow program access with a single bit of a chromosome. Every digit in a genotype can be represented by a binary bit instead of two bytes. So for the 45-bit chromosome, only three bytes, which include 48 binary bits, rather than 90 bytes are needed. The definition of the bit field structure is as below:

```
typedef struct tagGeneFields
{
    //First 16 Bits
    unsigned short X1 : GENESIZE_X1;//6
    unsigned short X2 : GENESIZE_X2;//6
    unsigned short PHID : GENESIZE_PHID; //4

    //Second 16 Bits
    unsigned short HAP : GENESIZE_HAP;//2
    unsigned short RHO : GENESIZE_RHO;//4
    unsigned short ALPHA : GENESIZE_ALPHA;//3
    unsigned short BETA : GENESIZE_BETA;//7

    //Remain 13 Bits
    unsigned short MODULE : GENESIZE_MODULE;//6
    unsigned short Z1 : GENESIZE_Z1;//7
}GENE_FIELDS
```

This bit field structure is used in the following template class definition (because the definition is very long, most of the program lines are ignored):

```
template <size_t _N> class mybitset {
    typedef unsigned long _Ty;
public:
```

```

:
public:
    enum {_Nb = 8 * sizeof(_Ty),
         _Nw = _N == 0 ? 0 : (_N - 1) / _Nb};
    union{
        _Ty _A[_Nw + 1];
        GENE_FIELDS GeneFields;
    };
private:
    :
};

```

In this template class definition, a union has been used in above template class definition. A union is a memory location that is shared by two or more different variables, generally of different types. In above list, the bit field variable *GeneFields* shares the same memory with the unsigned long integer variable *_A*. The variable *_A* provides the storage places for the chromosome, while the variable *GeneFields* provide a convenient way to retrieve the encoding value from every gene.

This template class has an argument *_N*, which means that the template class describes an object that stores a sequence of *_N* bits. In this gear design application, the chromosome length is 45, so the chromosome is defined in the program using the template class by following C++ statements (In the actual C++ program, these statements are not adjacent):

```

const CHROMOLENGTH = 45;
typedef mybitset<CHROMOLENGTH> CHROMO;
chromo = new CHROMO[Population];

```

In this way, retrieving the genotype value of all the nine gear design variables is straightforward as shown below:

```

int TempVal;

```

```

CHROMO* chromo1;
TempVal = chromo1->GeneFields.MODULE;
TempVal = chromo1->GeneFields.Z1;
TempVal = chromo1->GeneFields.ALPHA;
TempVal = chromo1->GeneFields.HAP;
TempVal = chromo1->GeneFields.X1;
TempVal = chromo1->GeneFields.X2;
TempVal = chromo1->GeneFields.BEATA;
TempVal = chromo1->GeneFields.RHO;
TempVal = chromo1->GeneFields.PHID;

```

Using this bit field structure for the definition of chromosomes, not only the memory storage is saved and the execution speed is improved, but also the program source code is more concise.

5.3.4 Variable Dimensional Problems

In the usual optimisation task a fixed number of variables corresponds to the number of degrees of freedom of the system modelled by the objective function, i.e., the model is fixed with respect to the structure of the problem. In practical application, however, some of parameters are sometimes given, so their values remain constant during the evolution of genetic algorithm, which means they are not variables any more. In other words, the variable dimension of the application has been changed. This kind of application is called variable dimensional application.

For the gear optimisation application, it is not always necessary to select all of the 9 variables at the same time, and therefore, this gear optimisation application is a typical variable dimensional application.

The most common way to tackle this variable dimensional problem is using a fixed set of genes which is combined to form a nine-gene genome, as shown in Table 5.2. Even though h_{ap} , α and m are unselected, i.e. they have fixed values, it is the whole chromosome of 9 gene segments rather than part of them that undertakes the

evolutionary operations. The corresponding genes of the unselected variables are involved in the evolution, but their values are still kept at the given values in decoding, instead of being mapped from the genome undertaken evolutionary operations. That may causes redundant gene segments and mapping deceiving.

Table 5.2 A chromosome with all 9 gene segments

Design variable	x_1	x_2	ϕ_d	$h_{ap} = 1$	g_{16}	$\alpha = 20$	β	$m = 5$	z_1
Number of bits	6 bits	6 bits	4 bits	2 bits	4 bits	3 bits	7 bits	6 bits	7 bits
Position of bits	0-5	6-11	12-15	16-17	18-21	22-24	25-31	32-37	38-44

In order to resolve this redundant gene segments and mapping deceiving problem, this study puts forward a new concept of dynamic and variable length chromosome. This dynamic chromosome only consists of the gene segments of the selected design variables. For those unselected design variables, their corresponding gene segments will not be included in the dynamic chromosome and, consequently, these gene segments will not be involved in the evolution process. In this way, the redundant gene segments and mapping deceiving problem can be overcome.

As described in Section 5.3.3.1, a chromosome is defined by a template class, which is impossible to be defined dynamically. Therefore, the definition of the chromosome remains unchanged in this approach. The dynamic chromosome is implemented by using a dynamically created mapping array. The length of the mapping array is equal to the sum of bits number of all selected gene segments. Also the value of each array element is a record of an available bit position. For example, for the chromosome shown in Table 5.2, x_1 , x_2 , ϕ_d , ρ_{fp} , β and z_1 are selected variables, and the sum of bits number is 34. The available bit positions are: 0-16, 18-21, 25-31 and 38-44. Therefore, the mapping array is an integral array of 34 elements. The content of the mapping array and its relation to the chromosome is shown in Table 5.3.

At the beginning of the program, the dynamic integral array is created according to a user's variable selection. During the evolution, the program will loop through this mapping array to map the operations to the correct bit position of the chromosome.

Table 5.3 Dynamic mapping array and its values

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	18
Index	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Value	19	20	21	25	26	27	28	29	30	31	38	39	40	41	42	43	44

The dynamic mapping array *m_ActiveGeneArray[]* is defined as a class member variable and initialised inside the *optimisation_setup()* function as follows:

```
void CGearOpt::optimisation_setup()
{
    :
    // Build up m_ActiveGeneArray to map active genes
    pos = 0;
    for(n = 0; n < NUMGENES; n++)
    {
        if(Gene[n] == 1)
        {
            for(i = 0, j = GenePositionArray[n]; i < GeneSizeArray[n]; i++, j++)
                m_ActiveGeneArray[pos++] = j;
        }
    }
    m_ActiveGeneSize = pos;
}
```

The variable *m_ActiveGeneSize* represents the length of the dynamic mapping array, which is used for loop control during the evolution operations. The following listing is part of the C++ programs for mutation operation, which explains how to use the mapping array in the evolution operations:

```
void CGearOpt::mutation(int tier)
{
    :
    do
    {
```



```

MutBit = int(rand() * m_ActiveGeneSize / AND_MAX);
TotRand += MutBit;
if (TotRand < m_ActiveGeneSize)
    chromo1->flip(m_ActiveGeneArray[TotRand]);
}
while (TotRand < m_ActiveGeneSize);
:
}

```

5.4 Multiple Objective Optimisation

5.4.1 Definition of Multiple Objective Optimisation

Most realistic optimisation problems, particularly those in design, require the simultaneous optimisation of more than one objective function. For example, gear design requires simultaneous optimisation of minimum weight and maximum strength. In the multiple objective cases, it is unlikely that the different objectives would be optimised by the same alternative parameter choices. Hence, some trade-off between the criteria is needed to ensure a satisfactory design. This kind of optimisation is the so-called multiple objective optimisation.

As described in [100], a general constrained multiple objective optimisation problem can be defined as in equation (5-32)

$$\text{Minimise } F(X) = \{f_1(X), \dots, f_i(X), \dots, f_m(X)\} \quad (5-32)$$

Subject to $X \in D$

$$D = \{X : g_j(X) \leq 0, \quad j = 1, \dots, J, \quad h_k(X) = 0, \quad k = 1, \dots, K\}$$

where X is an $n \times 1$ variable vector, $F(X)$ is an $m \times 1$ vector of objectives that are at least partly conflicting, $g_j(X)$ is the j -th inequality constraint and $h_k(X)$ is the k -th equality constraint. The set of design vectors that satisfies both equality and inequality constraints constitutes the feasible domain D .

Mathematically, a design solution $X^* \in D$ is said to be Pareto optimal if there does not exist another solution $X \in D$ such that $f_i(X) \leq f_i(X^*)$ for all $i = 1, \dots, m$ with strict inequality for at least one i . Any other feasible solution $X \in D$ with $f_i(X^*) \leq f_i(X)$ for all $i = 1, \dots, m$, is an inferior solution, also known as efficient, non-dominated solution. Multi-objective optimisations are actually to retrieve rational solution in multiple Pareto solutions, i.e. inferior solutions.

In this gear design application, X represents for a 9×1 vector consisting of 9 design variables as described in Section 5.2.2.1. $F(X)$ is a 6×1 vector consisting of 6 optimisation objectives as shown in equation (5-1)-(5-6). D represents the feasible domain which satisfies all constraints defined by equation (5-6)-(5-31).

5.4.2 The Weighted Sum Solution

5.4.2.1 The Weighted Sum Approach.

Decision makers in gear design are faced with a multiple objective optimisation problem when considering different and often-conflicting groups of interests and their demand for resources. When GAs is applied to single objective problems, the function value is used directly to determine the quality of individuals. However, in multiple objective problems, the quality comparison between different candidate solutions is not that simple. In implementing GAs for multiple objective problems, one must decide on how to assign fitness to the individuals (based on multiple function values) and how to maintain diversity in the population to avoid premature stagnation.

In this study, the weighted sum approach is utilised to turn the multiple objective problem into a single scalar objective problem, whose solution is a Pareto optimal point for original multiple objective optimisation, by using a weighted sum of the different objective functions. According to weighted-sum-based optimisation, the fitness for this problem is transferred into a scalar function through the equation (5-33):

$$F(X) = \sum_{i=1}^6 w_i f_{scaled-i}(X) \quad (5-33)$$

where $f_{scaled-i}$ is objective f_i normalised to [0,1] by equation (5-34) and $W = (w_i)_{i=1,5}$ is the weights vector where the sum of the weights equals 1. It is easy to prove that the minimiser of this combined fitness function using the weighting sum approach is Pareto optimal [100].

A major difficulty lies in the setting of the weights in terms of the relative importance of the objectives especially where results are particularly sensitive to the weighting ratio. Another task for obtaining the fitness is to determine the maximum and minimum to normalise the individual objective fitness values. Since the maximum and minimum are not known beforehand, both are determined at each generation and varied during the implementation of the algorithm.

In addition to the weighted approach, other techniques such as Homotopy techniques, Goal programming, Normal-boundary intersection and multilevel programming have been developed for the multiple objective optimisation [104].

5.4.2.2 Fitness Normalisation

In order to calculate each normalised objective function value in equation (5-33), f_{good} and f_{bad} are used to scale the objective functions, as shown in equation (5-34) to unify the maximising and minimising problem into a common formula. Therefore all objectives are of the same order of magnitude and also convert the problem to a maximisation type.

$$f_{scaled} = \frac{(f - f_{bad})}{(f_{good} - f_{bad})} \quad (5-34)$$

If an objective is to be maximised (minimised), then f_{good} would be greater (smaller) than f_{bad} . Therefore the scaled objective function values (f_{scaled}) lie in a range of [0, 1], and the greater the value of the scaled objective function, the better the solution is.

This function is applied in conjunction with the deviation in fitness, thus providing greater resolution to the selection roulette wheel. The effect of this is to define a

noticeable ranking order without encouraging super convergence, caused by excessively biased scaling of fit genomes.

5.4.2.3 Multiple Objective Optimisation Results of Gear Design

How to choose appropriate weighting factors depends on the users' design purpose. In the gear design application, for example, when both the centre distance and the contact stress are used as optimisation objectives, the weighting factor for the centre distance and the contact stress depends on importance of space size and strength. If the space size is more important than strength, then the centre distance should have a greater weighting factor, and vice versa. But normally a trade-off has to be done between them when both are taken into account for the design purpose.

The overall Pareto solutions for the centre distance and the contact stress are obtained by changing the two weighting values, as shown in Figure 5.3. When the weighting value for centre distance is set 100 percent and the one for contract stress is 0 percent, the centre distance, as the main objective, reaches its minimum mean while the actual contact stress reaches its maximum, and vice versa. In order to consider both the two controversial objectives, a trade-off point between the two extreme situations could be achieved, hence the designer can choose an appropriate one out of these solutions.

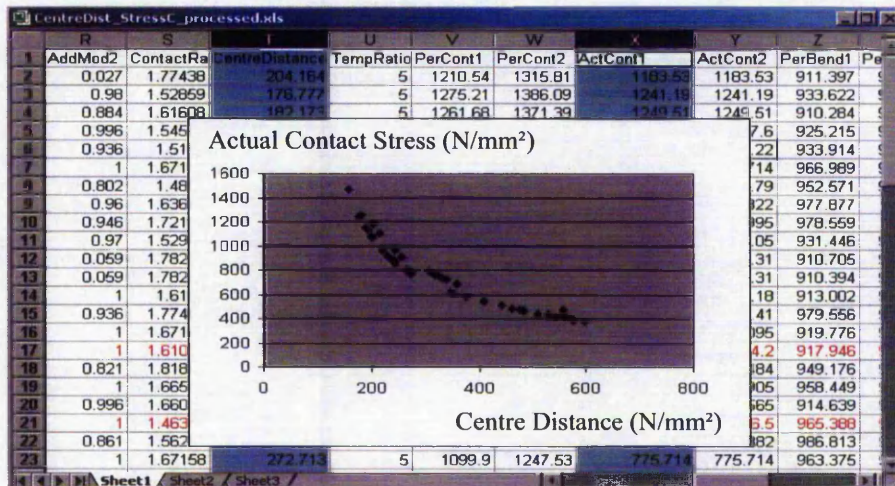


Figure 5.3 Pareto solutions of centre distance and actual contact stress

Each point in the front graphics represents a pair of values of centre distance and contact stress, and the corresponding design solution parameters are recorded in the Excel sheet.

Likewise, changing the weighting values of the actual contact stress and actual pinion bending stress yields the entire Pareto solutions of actual contact stress and actual pinion bending stress, as shown in Figure 5.4. In this case, minimising the bending stress difference between pinion and wheel is also used as a design objective and its weighting value is set to 100 percent and kept constant.

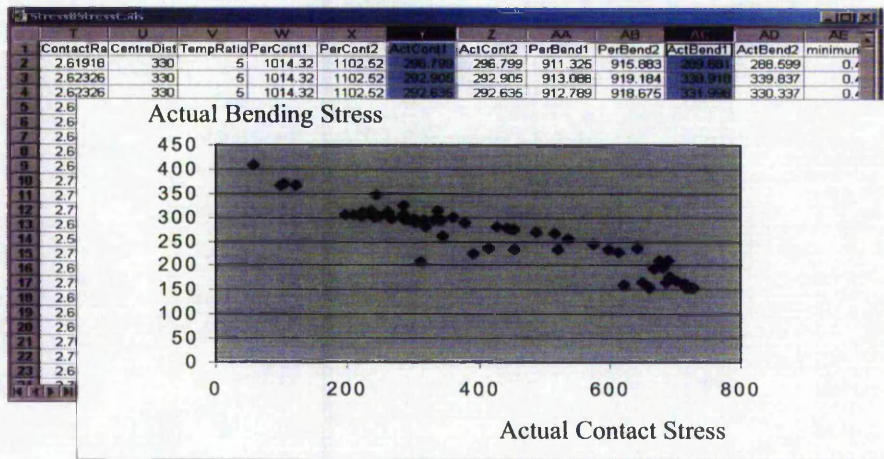


Figure 5.4 Pareto solutions of actual contact stress and actual pinion bending stress

The Pareto solutions as shown in Figures 5.3 and 5.4 help designers to choose a final solution from a wide range of feasible solution collections.

5.5 Variable Penalty

Gear design involves many design constraints problems from different domains. In this study, a variable penalty approach for dealing with the constraint violation is investigated and presented in this section.

It was found that tuning of the penalty function dramatically affected the process of convergence and the quality of the result [103]. If a set of penalties is too harsh, then the few solutions that do not violate constraints will quickly dominate the mating pool

and yield sub-optimal solutions. A penalty that is too lenient can allow infeasible solutions to flourish as they can have higher fitness values than feasible solutions. Often, the algorithm must be rerun several times before a combination of penalties is found that allows infeasible solutions to die and feasible solutions to flourish. The main difficulty in applying penalty functions is that they are problem dependent.

In this study the idea of Simulated Annealing [105] is employed to construct a variable penalty function that is tuned during the genetic algorithm process [100]. In the early stages of the algorithm, infeasible solution kept in the population can be important to the genetic process as an aid to searching for the globally optimal solution. Since the cooling temperature function attenuates during a given run, so the penalty gets gradually harsher to make it possible to finally come closer to the global feasible solution. On the other hand, the penalty values are not related to the weighting of each constraint factor, even though they are related to the amount of each constraint violation and time of the convergence process.

5.5.1 The Variable Penalty Function

In the proposed approach, the simulated annealing idea is employed to cope with constraints to provide a variable fitness function, which is related to the constraints at hand and the temperature schedule used in simulated annealing algorithms. The variable fitness function is calculated in equation (5-35).

$$\Phi(X,t) = \alpha(M,T)F(X) \quad (5-35)$$

The final fitness function $\Phi(X,t)$ equals the product of $F(X)$ and $\alpha(M,T)$. $F(X)$ denotes the normalised objectives (as elucidated in detail in (5-32) - (5-34) in Section 5.4), to be maximised, which must have a positive (or zero) value throughout its domain. $\alpha(M,T)$ is the attenuation factor, which depends on two parameters, M and T . M ($M \geq 0$) is the measure of constraint violation and equals zero in the case of no constraint violation. T is the temperature schedule parameter, which is a function of the running time of the algorithm, and the initial temperature parameter. The following function has been chosen for the attenuation factor since it has the required properties.

$$\alpha(M,T) = \exp(-M/T) \quad (5-36)$$

As execution proceeds, T tends to 0 (or small values), and then α tends to 0 as well and hence, by equation (5-35), the fitness tends to zero too. This means that the individual is penalised harshly and should be excluded from the populations at the end of a run. In contrast, at the beginning of the algorithm, T is large and $\alpha \approx 1$ in order to make the penalty of constraint violation less harsh and utilise infeasible states as needed to find the global maximum. The calculation of T is presented in Section 5.5.2, and M in Section 5.5.3.

5.5.2 The Temperature Schedule for Variable Penalty

$T(t)$, used for calculating the attenuation factor, denotes the cooling schedule from high temperature to lower temperature at a time t (a generation). It can be described in one of the following ways:

$$T(t) = T_0 / \log(t+1) \quad (5-37)$$

$$T(t) = T_0 / (t+1) \quad (5-38)$$

$$T(t) = T_0 / \sqrt{(t+1)} \quad (5-39)$$

where T_0 is the initial iteration temperature.

For these three schedules, $T = T_0 / \log(t+1)$ is most generally used.

5.5.3 Constraint Calculation

In this application, the value for the amount of constraint violation is calculated by the following equation:

$$M = \frac{\sum_{j=1}^J \max(g_{j,i}(x), 0) + \sum_{k=1}^K |h_{k,i}(x)|}{\left(\sum_{i=1}^{popsize} \sum_{j=1}^J \max(g_{j,i}(x), 0) + \sum_{i=1}^{popsize} \sum_{k=1}^K |h_{k,i}(x)| \right)} \quad (5-40)$$

where $g_{j,i}$ is the j -th inequality constraint value given by equation (5-7) - (5-22) for the i -th individual, and $h_{k,i}$ is the k -th equality constraint value for the i -th individual.

M is one of the two parameters used for the attenuation factor calculation in equation (5-36).

The initial estimate for the temperature parameter T , i.e. T_0 , depends on the actual application being considered and is determined by trial and error. As a rule of thumb, the starting temperature can be estimated to be the same value as the mean constraint violation \bar{M} .

5.5.4 Instances of Calculation and Results Analysis

Many instances of optimisation utilising this variable penalty approach are illustrated in this section. The results of these instances are used not only to verify the performance of variable penalty approach, but also to analyse the influence of the temperature schedule and the initial temperature.

The temperature schedule and the initial temperature parameter, in the above-presented variable penalty function, have a crucial influence on the optimisation process and the results. How to select effective temperature schedule and initial temperature parameter is a key technique to the variable penalty approach. The analysis of the results in this section is helpful for a user to make a correct choice.

All of the results in this section are based on the following gear design specifications: Power 100kW, speed 960 rpm, speed ratio 5, maximal helix angle 35° , gear hardness 825 HV. The algorithm running parameters were: Population 2000, crossover rate 0.8, and mutation rate 0.05

5.5.4.1 Effectiveness of Variable Penalty Approach

To verify the effectiveness of variable penalty approach, the convergence profiles of 10 optimisation instances are demonstrated in Figure 5.5. The conditions of this experiment are list below:

- Temperature schedule: $T(t) = T_0 / \log(t + 1)$
- Initial temperature: $T_0 = \bar{M}$
- Objective: Centre distance

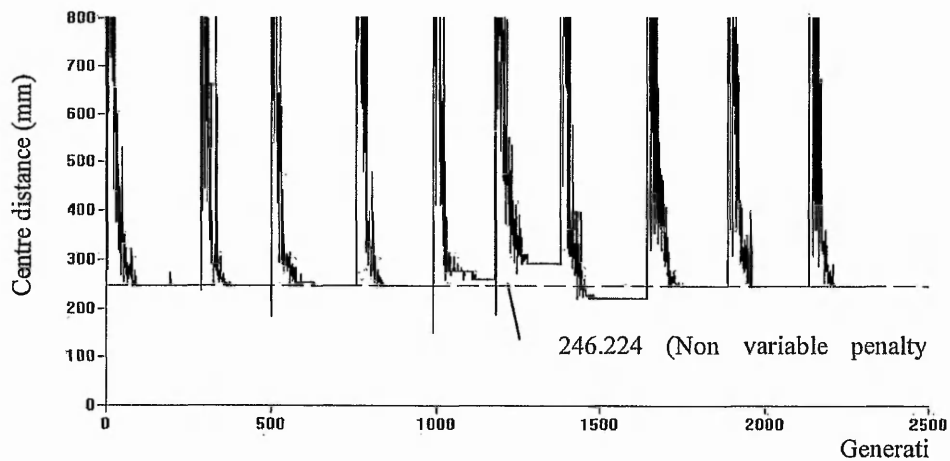


Figure 5.5 Variation of the calculated centre distance with generation, ten runs

The centre distances calculated from 10 runs are: 246.336, 246.336, 246.224, 246.336, 261.396, 293.725, 221.626, 246.329, 246.336, 246.324. These results are compared with the value 246.224 calculated, for the same example, by using a conventional non variable penalty approach, where the constraints are handled according to the formulas given above, but with a constant weighting factor for each constraint. Seven of them are closely similar to the value determined from the non variable penalty approach, which is represented by the dashed line.

Figure 5.6 is an enlarged portion of the first experiment, which shows that a reasonably close estimate to the optimum value is obtained in just under 100 generations, and that there is no further change in the estimated value after about 150 generations.

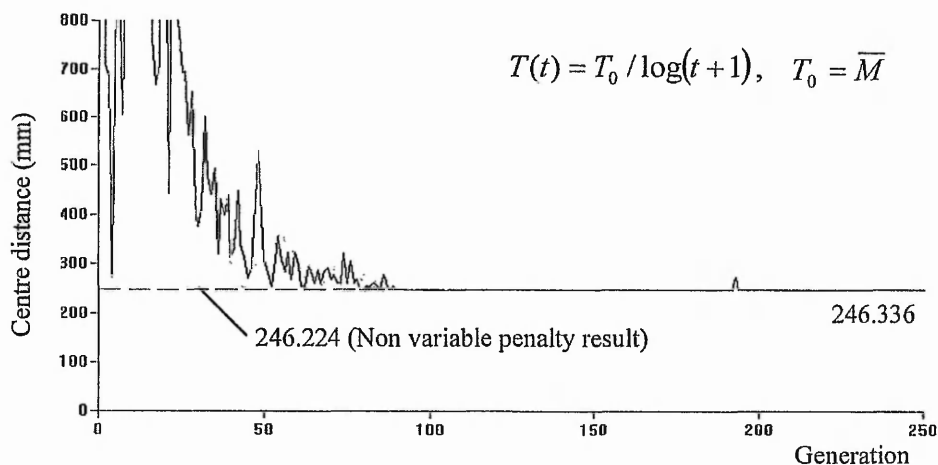


Figure 5.6 Variation of the calculated centre distance with generation, one run

5.5.4.2 Effect of Temperature Schedule

The influence of different temperature cooling schedules has also been investigated. The most common alternatives from the literature, equations (5-37) and (5-38), have been used for investigation. \bar{M} , which is automatically calculated by program to every generation, is still used as the estimate for T_0 . Figure 5.7 shows the convergence profile for the centre distance using (5-38). It can be seen that convergence occurred more quickly than with equation (5-37), comparing with Figure 5.6. The average value of \bar{M} for 10 runs was 48.23, and the average centre distance was 273.272.

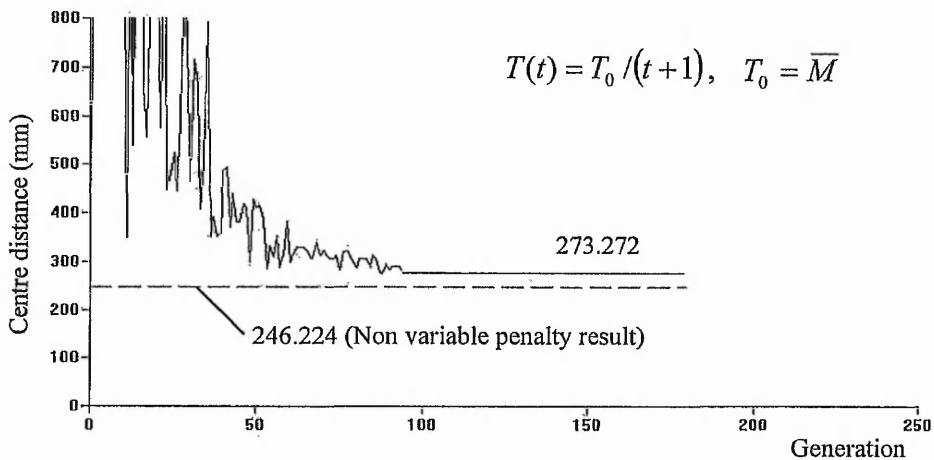


Figure 5.7 Effect of varying the function for the cooling schedule

5.5.4.3 Effect of Initial Temperature Parameter

To verify the effect of the initial temperature parameter, the alternative approach of setting a fixed value for T_0 has also been studied. The temperature schedule function used for this study is still $T(t) = T_0 / (t + 1)$, i.e. the same as the function used in Figure 5.7, but the initial temperature parameter T_0 is no longer \bar{M} . Several values have been tested for T_0 , and the results are used for comparison in Figure 5.7. For $T_0 = 100$, there was no significant difference, but setting $T_0 = 1000$, the convergence value for the centre distance is lower than for the previous two cases, and closer to the non variable penalty result, as shown in Figure 5.8. There is no further improvement in the optimal estimate for the centre distance on setting $T_0 = 10000$.

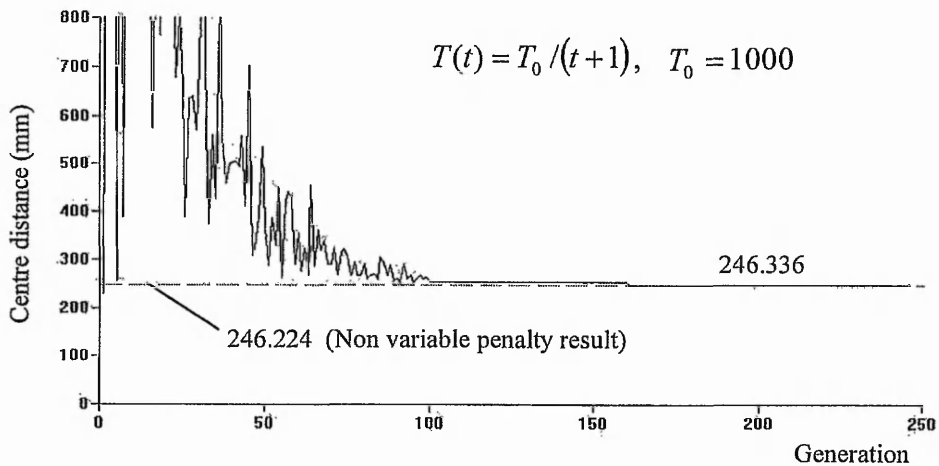


Figure 5.8 Effect of varying the initial temperature parameter

It can be seen that the results obtained using equation (5-37) were similar to those for (5-38). It would appear therefore that it is better to manually adjust the value for T_0 , using the value of \bar{M} as the initial estimate.

5.5.4.4 Effect of Fixed Temperature

As a further test of the effectiveness of using equations (5-37) - (5-39) to adjust the cooling schedule, some results were obtained with constant values for T of 100, 500 and 1000. Figure 5.9 shows the resultant profiles. It can be seen that using $T = 100$, gives the most rapid convergence to a stable constant value, which is also close to the non variable penalty result. However, in this case the converged values for the centre distance are much more strongly dependent on the selected value of T than those obtained using any of the time dependent estimates for T , i.e. equations (5-37) - (5-39). Thus, in this case it is preferable to use any of these schedules, rather than using an arbitrary fixed value for T , which may produce inappropriate results.

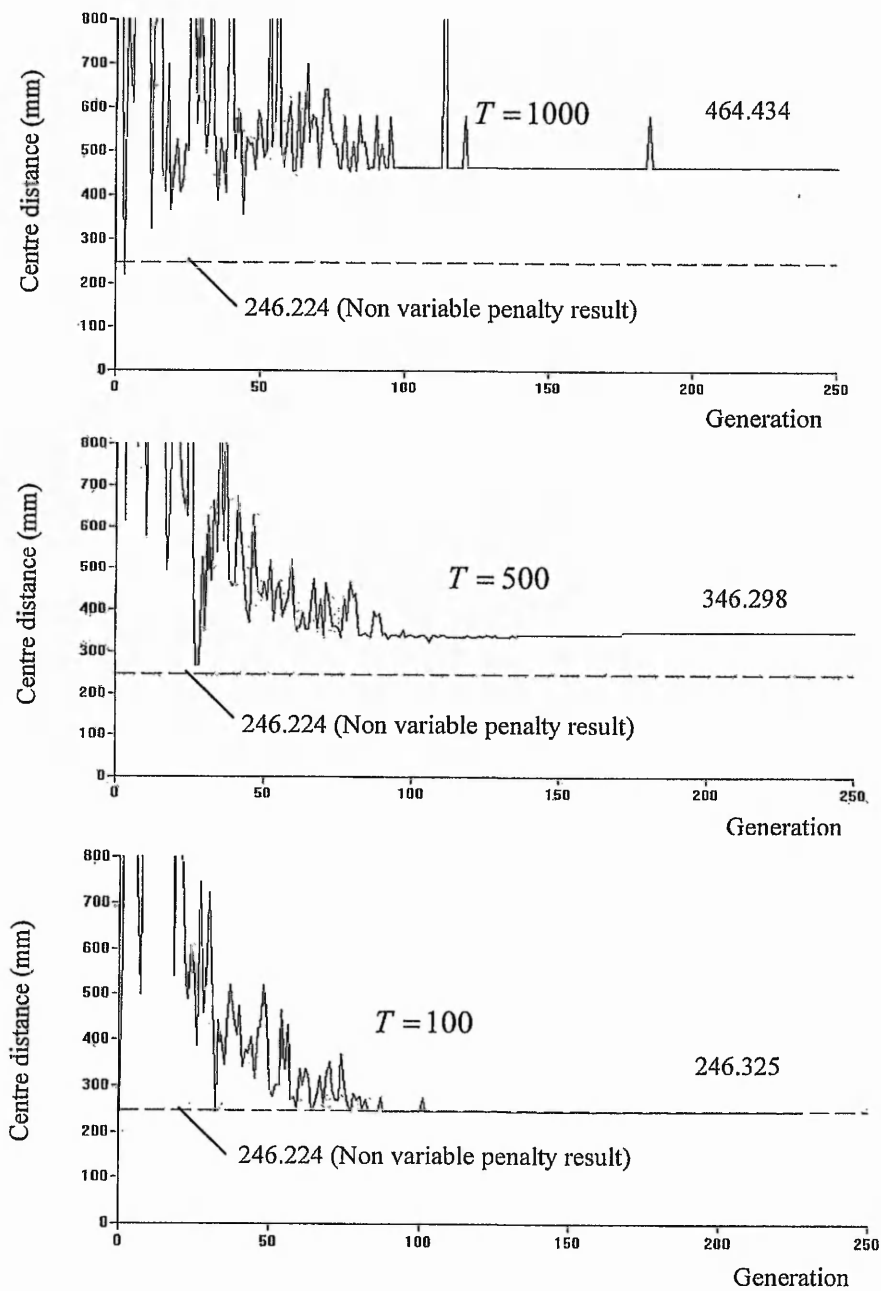


Figure 5.9 Effect of using a fixed temperature on the convergence process

5.5.4.5 Cascaded Structure

The effects associated with using the two tier cascaded structure, rather than just a single tier, are considered in this section. Taking the results obtained at the first stage as the warm value for the second stage, a narrower area around the warm value is searched in order to obtain a more precise solution. Figure 5.10 shows examples of the

results obtained using this approach. It was found that the cascaded method gave more precise results, with very few exceptional instances.

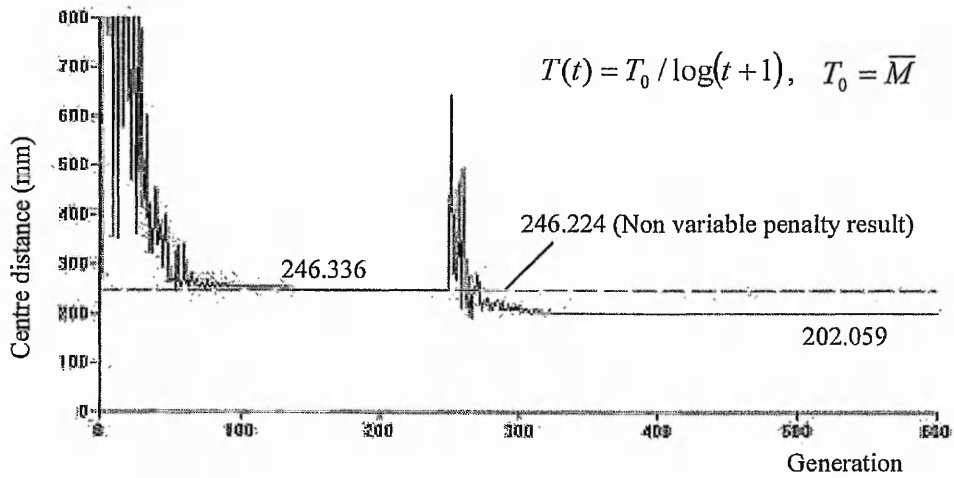


Figure 5.10 Convergence process for two tiers cascaded structure

The results from the cascade structure GA and variable penalty method are compared with that from non cascade structure GA and non variable penalty method, as listed in Table 5.4. In this case a cascade structure GA, with $T = T_0 / \log(t + 1)$, $T_0 = \bar{M}$, and two tiers were implemented. It was found that the results for the centre distance are improved because of the cascade structure.

Table 5.4 Result comparison of cascade and non cascade structure

Results	Non cascade structure and non variable penalty	Cascade structure and variable penalty	
		First tier	Second tier
a (mm)	246.224	246.336	202.059
ϕ_d	1.399	1.399	1.099
m (mm)	2.5	2.5	2.5
h_{ap}	1.25	1.25	1.25
α (Deg.)	20	22.5	22.5
β (Deg.)	4.271	4.166	3.419
ρ_{JP}	0.2596	0.3496	0.3496
x_1 (mm)	0.238	0.224	0.119
x_2 (mm)	-1	-0.936	-0.436
z_1	33	33	27

B (mm)	115.739	115.723	74.3148
ε	3.27464	3.09328	2.5638
$\Delta\sigma_b$ (%)	0.06	5.507	0.034
ΔS_r (%)	0.17094	0.001185	0.612393

5.6 Summary

An optimiser is designed to help a designer to perform multiple objective design optimisation based on the distributed system described in Chapter 5. This chapter presents the key issues for designing the optimiser.

First of all, the gear design optimisation model, which includes 9 design variables, 6 design objectives, and 25 design constraints, is defined based on addendum modification design. To help with the understanding and implementation of this model, basic concepts and general knowledge about addendum modification in gear design is introduced briefly.

Next, the genetic algorithm implementation utilised for gear design optimisation is described. The basic concept of a genetic algorithm, i.e. the general procedure of genetic algorithms and operations on chromosomes, is introduced. In order to improve the performance of traditional genetic algorithms, some innovative approaches, including cascade structure, bit field representation of a chromosome, and dynamic mapping method of variable dimension problem, are fulfilled. The cascaded structure with two tiers is used for the whole optimising process to obtain higher efficiency of optimisation and more precise results. The C++ Bit field structure in conjunction with union data type is used to provide a highly efficient data structure for chromosome representation, which simplifies the transformation calculation between binary data and encoding value so that the running efficiency of the program is greatly improved. A dynamic mapping array is created to obtain the dynamic combination of genes according to a user's variable dimension requirements.

Furthermore, multiple-objective optimisation is studied and the weighted sum solution is given. Changing the weighting factors is employed to obtain the entire collection of

solution for a user to select a final solution from a wider region. The methods of choosing appropriate weighting factors are discussed and, the gear optimisation results obtained by applying these methods are illustrated.

Finally, a variable penalty function based on a simulated annealing algorithm is constructed to deal with the constraints, which provides another approach to tune the process parameters of convergence instead of tuning the penalty function weighting coefficients. Many instances of optimisation utilising this variable penalty approach are illustrated and the results of these instances are used to verify the performance of variable penalty approach and analyse the influence of the temperature schedule and the initial temperature parameter.

Chapter 6 Remote Invocation of Single Large Size of Program

6.1 Introduction

In the engineering design domain, there are many large-size computing programs. Most of these computing applications deal with much parameter input and output data, and have time-consuming execution. For these programs, basically it is ideal to be located and executed on the owner's computer with response to the client request to make sure thin-client requirement. More importantly, such time-consuming programs should not be interrupted during their execution. Furthermore, in a collaborative environment, it is probably better to make such a large-scale and time-consuming program to be invoked singularly, otherwise the collaboration with other partners may mean waiting for too long a time to retrieve the results.

Therefore a Web-based system should be designed to support singular large-scale program execution without interruption due to network connection, and to provide an interface for a user to do convenient parameter input and result visualisation.

The typical application style for this problem is to let a Web server process the user input, run the program and serve dynamic contents in response to client requests. In order to do this several methods have been devised such as Common Gateway Interface (CGI) and Java Servlet. CGI programs can be written with different languages such as C/C++ or Perl, while the Java Servlet is written in the Java language only.

The speed problem caused by the CGI itself, however, has been explored and has affected the package's utilisation for multiple users. In order to resolve the problem,

another technology for Web server extension, Java Servlets, is chosen as a solution and implemented in this PhD project. Java Servlets are used not only to improve the performance in the multi-user situation, but also to provide a greater capability when combined further with CORBA.

A servlet is a small piece of Java code that a Web server loads to handle client requests. It is a platform independent Java sever-side module that fit seamlessly into a Web server framework and can be used to extend the functionalities of a Web server. Servlets receive a request from a client, dynamically generate the response, and then send the response containing an HTML or XML document to the client. This provides more forms of dynamic computing results to the client request. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases, which simplify rapid application development. Therefore Servlets are ideal to provide server-side services such as accepting data from users, writing data to or retrieving from databases, and returning data in dynamic pages.

Applets are client-side of Java code that can be inserted in a HTML page. They are downloaded along with Web pages on a client machine. Applets are used in this application in combination with Servlets, in order to provide graphical resultant data.

Servlets can also use CORBA to access an application server and further invoke an application object that is located on the same machine with few possibilities to interrupt. Programs ported on a Web server might be written in different languages and therefore CORBA can be used to establish a bridge between Servlets and application programs.

From the above description it can be inferred that Applets, Servlets and CORBA are used in the remote execution of a single large size program. A Servlet makes it possible to change the front end for the application entirely to HTML and thus it can be used for sending and retrieving complex data. The large size of program might be a

non-Java application and then CORBA is the most effective mechanism to establish the communication between a Web server and application server. Applets are used for demonstrating the resultant data in graphics. To design such a system, a Web server-centralised model can be considered to construct the architecture, as described in Chapter 3.

In this chapter, the process for implementation of the Web-based design system using Java Applets, Java Servlets and CORBA is described. The utilisation of other relevant techniques such as HTML, JavaScript, Web server and protocol are also presented. In addition, an example is provided to demonstrate how these techniques are applied in this application.

6.2 Internet Solution for Executing a Singular Large Program

6.2.1 Standalone Design Application Package

The package is used to optimise the design of external spur and helical gears with involute tooth profile. The original gear design software was developed in Visual BASIC and C++. It is a Windows platform specific executable program and consists of the following two parts:

- A friendly graphical user interface (GUI)
- An algorithm program

The GUI was developed using Visual BASIC while the algorithm program was written in C++. The GUI is used for design data input, setting-up optimisation specifications (goals, weight factors, population size and number of tests), and displaying the results. The algorithm program conducts the design optimisation. Both of parts are fully integrated into a single software environment.

Up to nine gear design parameters can be optimised, including tooth face width, module, pressure angle, helix angle, rack tip radius, addendum coefficient, two addendum modification (tooth profile shift) coefficients, and number of pinion teeth.

There are six optimisation objectives including minimizing face width, minimizing centre distance, reducing the difference in tooth root bending stresses between the pinion and wheel, reducing the difference in tooth tip specific sliding ratio between the pinion and wheel, minimizing bending stress at the tooth root and minimizing contact stress at the pitch circle.

Some constraints from motion requirement, manufacturing limitations and cost consideration are considered during the optimisation, including that the maximum tooth root bending stress cannot exceed the allowable stress, tooth contact stresses cannot exceed the allowable stress, the sliding/rolling speed ratio cannot exceed 3, and so on.

The execution structure of the software is shown in Figure 6.1. The GUI is used for the user to input the necessary information including application conditions (power to be transmitted, speed ratio, etc.), algorithm running parameters (individual number, test number, etc.) and requirement parameters (quality, life, and so on). These values are then written into the input files. The Visual BASIC program then calls the algorithm program to perform the design optimisation. This first reads the data from the input files and then starts the algorithm procedure, with searching, evolving and obtaining rational solution out of the design space. After the execution is completed, the results are written into output files, which are then displayed to the user.

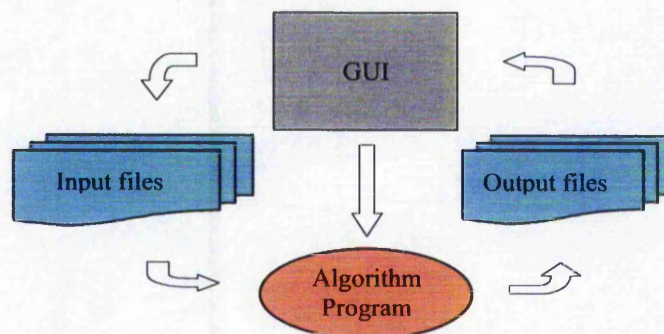


Figure 6.1 Structure of the original gear optimisation software

The optimisation process may be time-consuming because the multiple parameter design problem is computationally expensive. The time it takes to execute depends on the given size of the population, the given number of tests and the selected number of parameters to be optimised.

6.2.2 Internet Solution

The gear design optimisation could not be conducted on the Internet unless the following problems have been resolved:

- How to remotely invoke a large sized software package over the Internet.
- How to pass the user's inputs data to the executing package and to send the results back to the user.
- How to allow multiple users to run the package at the same time.
- How to allow users to implement the invocation of software program from any machine over the Internet.

In addition, the copyright and security problems for the package have to be considered.

To accomplish this, in this application, the combined module of Java Servlets, Java Applets and CORBA is used as an alternative of the CGI. The complete structure of the system is shown in Figure 6.2.

The gear optimisation package is located on the server side. A user on the client side wishes to conduct gear design by invoking remote design resources. What the user needs to do is only to interact with a Graphical User Interface (GUI) in a Web page from a Web browser.

A registered user could access the main page for design, after authentication by entering their username and password within a welcome page. The user could give all the optimisation parameters through the interface in the HTML page. Then the parameters are sent to the server which calls the server-side extension such as Servlets. When the user clicks on the submission button, a Servlet program, which is located on the server, is activated by the HTML code. It parses the data and writes them into the

input files and invokes a C++ algorithm program located on the server through the CORBA communication bridge. An applet along with a dynamic page downloaded on the client side is for the user to view the execution progress of the algorithm program. A flag data representing the status of program execution is produced by the program and retrieved by the applet procedure that keeps updating the state of progress bar. Therefore the user can view the execution progress state of the program. When the execution is completed and output files are created, Servlets would pass the results back to the client in the preferable form, in a data table in HTML file, graphics, excel data or XML data.

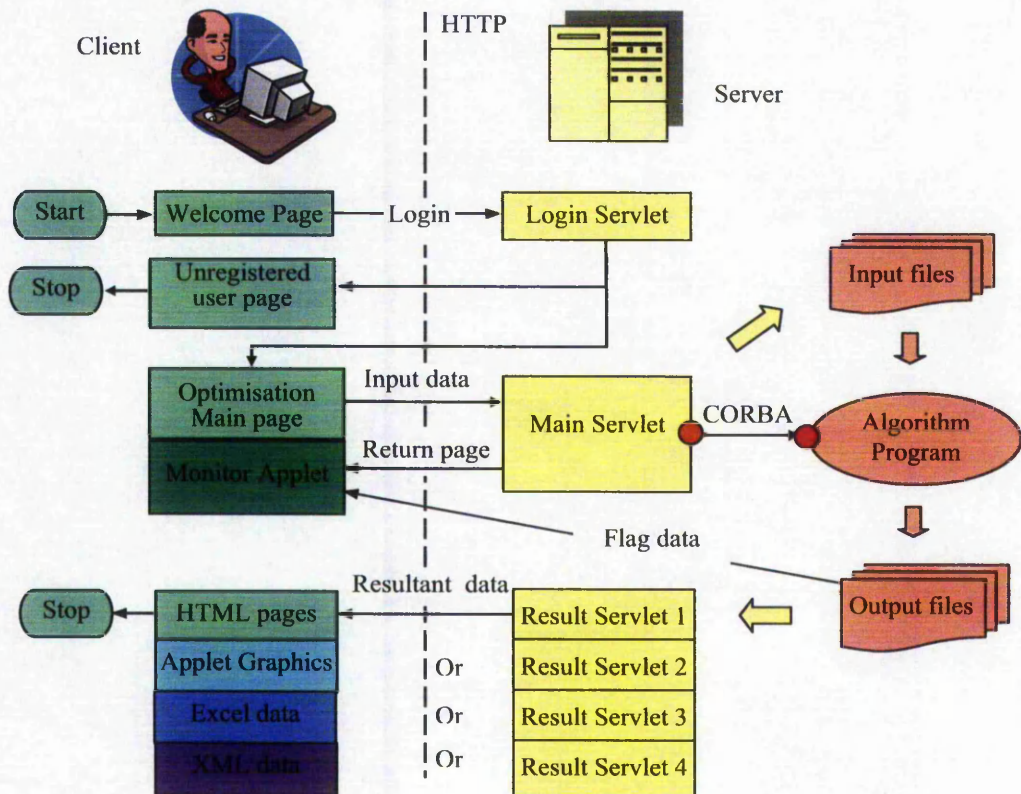


Figure 6.2 Hybrid architecture of the system

The gear optimisation package resides and runs on the server machine. This is considered to be a secure approach since the program is not downloaded to the user's machine and the client has no access to the source code. From the view of the user, there is no special requirement except for a Java-enabled Web browser.

The existing gear optimisation package is of large size, time-consuming and platform-dependent. The execution of it on the server depends only on the power and platform on the server-side, and is not relative to the client-side machine. On the other hand, HTML pages and even Applet along with them are platform- and operating system-independent. It is possible for users to run the package from any machine whether it is of UNIX or Windows system over the Internet.

A HTML file, instead of the Visual Basic program of the original optimisation package, is used to pass the data from the client to the server which in turn invokes the servlet programs to process them, to call the package and to send back the result to the user. The utilisation of the servlet program improves the speed in response to the client request, which will be further described in Section 6.3.2 in more detail.

The multi-thread feature of Servlets makes it possible to support multi-user requests at the same time. In order to support this, the gear design optimisation program has been modified. For example, a flag has been used as a symbol when the program is completed and outputs the status information during the running of the package, in order for the servlet programs to control it. User information, such as name and address, has been used as the running parameters in order for the servlets to manage multiple users. Further details will be described in Section 6.3.

6.3 Development of the System

6.3.1 HTML File

The user interface is written in HTML, which is platform-independent and the codes are interpreted locally, i.e., on the client's machine. The interface of the original gear optimisation software was created using Visual Basic and C++, which is platform-dependent and hence is not suitable for the Internet application.

Another important function of the HTML is its features called *form* that makes it possible to send information such as design parameters from the user to the server. After the user inserted the data and pressed the submission button, all the data within

the form will be transferred to servlets located in the server. The servlet programs then perform the processing and send the results back in response to the user. The parameter input form is shown in Figure 6.3.

The screenshot shows a web browser window with the address bar displaying `http://152.71.17.151/servlet/GearOptLoginServlet`. The page content includes a navigation menu on the left with links for Introduction, Information, Optimise, Results, and Help. The main content area is titled "Initial Design - Application" and contains a form with the following parameters:

Population Size	1000	Surface Safety Factor	1
Number of Tests	2	Maximum Helix Angle	35 deg
Power	100 kW	Load Distribution Factor	0
Input Speed	960 rpm	Gear Offset	60 mm
Speed Ratio	5		
Maximum Axial Force	0 N		

Figure 6.3 Parameter input form

JavaScript, included in the HTML, has been used to offer the HTML dynamical behaviour. Because HTML documents are static and plain-text files, they do not have the ability to perform any calculation or validity to check the data submitted to the server. To make the Web page more dynamic and user-interactive and to perform the data validation, JavaScript is used. JavaScript is a scripting language which can be embedded into the HTML code in a text format and interpreted and run by the Web browser whenever the user retrieves the Web page, and it does not need to be compiled into program.

The main task of the JavaScript used in this project is to check the data before passing them to the server. If there is an invalid data, i.e., any of the data being wrong, missing or out of the permitted range, the JavaScript would prevent transfer of the data to the server and display a message box informing the user what the error is. The JavaScript code is embedded into the HTML document using the SCRIPT tag. The user name and password have to be entered in the welcome page of the HTML file to be sent to the *login Servlet* before data input.

6.3.2 Servlets

6.3.2.1 Servlet Vs CGI

CGI, the dominant interface for extending Web servers for years, is defined to allow Web servers to process user input and serve dynamic contents, and to connect to the external program. CGI programs can be developed in any script or programming language, such as Perl, and C/C++. Because CGI support was built into every Web server on the market, CGI was a popular choice for development tools and applications that could add dynamic capabilities to a Web site. However CGI has drawbacks, which have been experienced during this project. A new process needs to be created for each request. This leads to performance problems at popular Web sites that handle requests from multiple users. In addition, this package requires significant time for execution.

Java Servlets are small, platform-independent server-side programs that also extend the dynamical function of the Web server. Servlets run on a Java-enabled Web server and can provide the Web service in the module of request-response. Generally they could implement the same functions as CGI, but they have their own properties. The main property of Java Servlets is to improve the speed issues of CGI. Servlets solve the performance problem by executing all requests as threads in one process. It starts a new thread (rather than a new process) with each client request.

The so-called *processes* are different programs, such as Word and Excel, running in the same computer system, which have different addresses and spaces. Context switching between different processes and changing currently running process are complicated. The communication between different processes is expensive and limited. Even though many processes can be running simultaneously, only one process can be communicated with.

A *thread* is a control stream with one order in a *process*, also called lightweight process. Threads share the same address and space and construct one big process. The communication between threads is simple and effective, and context switching is fast and is part of the whole program. Threads could be executed separately. Running several threads simultaneously in one process could be used to different tasks. Multiple

threads provide the interactive power of the program, more GUI and more powerful server functions.

The essential reason why the Java Servlets could have a better performance than CGI is that Servlets are multithreaded and run within the process of the Servlet server. A process context switch is not required to handle each Servlet request. When the first time that a servlet is requested, it is loaded into the Web server's memory space. Subsequent client requests for the servlet result in calls to the servlet instance in memory.

Servlets offer many other added benefits to the developer, including ease of development, fast throughput and response, inter-server communications, and all of the features inherent in Java.

6.3.2.2 Servlet Running Environment

A Servlet is a Java class and therefore needs to be executed in a Java VM by a service that is called a Servlet engine. The Servlet engine loads the Servlet class the first time the Servlet is requested, or optionally when the Servlet engine is started. The Servlet then stays loaded to handle multiple requests until it is explicitly unloaded or the Servlet engine is shut down.

Getting the Servlet engine is either by obtaining a Servlet-enabled server, which has a built-in Servlet engine, or by obtaining a Servlet engine add-on that will be added to a common server. Some Web servers, such as Sun's Java Web Server (JWS), W3C's Jigsaw and Gefion Software's LiteWebServer (LWS) are implemented in Java and have a built-in Servlet engine.

Other popular Web servers, such as Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS) and the Apache Group's Apache, require a Servlet engine add-on module. The add-on intercepts all requests for Servlets, executes them and returns the response through the Web server to the client. Examples of Servlet engine add-ons are Gefion Software's WAICoolRunner, IBM's WebSphere, Live Software's Jrun, New Atlanta's ServletExec and Apache's Tomcat.

In this project Apache Tomcat is chosen as Servlet running environment. Apache Tomcat is the Servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process [<http://jakarta.apache.org/tomcat/>].

All Servlet API classes used to create Servlets code and a simple Servlet-enabled Web server are combined into the Java Servlet Development Kit (JSDK), available for download at Sun's official Servlet site.

6.3.2.3 Implementation of Servlets

Servlets are some Java classes located on the server and can accept the request in a HTML file from a user. They retrieve the data in the format of a long string from the form and subsequently process the data and send the results back to the user. The following types of servlets are developed in this project:

- *Login servlet* is designed to check the user identity, including username and password, to ensure that only a legal user is allowed to access the optimisation package form the Web page.
- *User management servlets* are designed to maintain user information, and to add or delete the user from the list of the registered users. If a user's registration has expired, the servlet will delete the user name automatically from the list. This servlet is only accessible for a system administrator of the server other than for users.
- *Main servlet*. The main tasks of the servlet are: (1) to accept the request information by the specific protocol and to retrieve the data from the user, for which the data is sent via the HTML *form* using the POST method and is extracted using the *getParameter()* method; (2) to execute some necessary pre-processing calculations for the data using Java API; (3) to create input data files for the gear design optimisation using *FileWriter* and *PrintWriter* classes of Java; (4) to run the optimisation program, using the CORBA communication mechanism between Servlet client and program object, as described in detail in

Section 6.4; (5) to return a HTML page with a progress bar showing the optimisation progress, as shown in Figure 6.4. The progress bar is controlled by a Java Applet program, which is inserted into the HTML page returned by the servlet. It reads the status data produced by the optimisation program dynamically and displays the graphical process stripe which shows the progress of the execution.

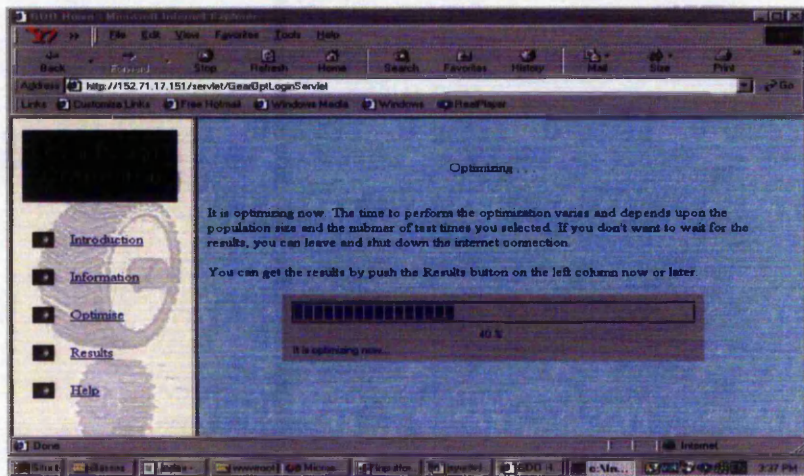


Figure 6.4 Progress bar of the remote program execution

- *Result servlets* When the user presses the button *Result* on the left hand side of the screen, if the optimisation program is not completed, then the progress bar is displayed, otherwise the result page, as shown in Figure 6.5, will appear on the user's screen. The result servlets, linked to the buttons *Result* and *Comparison* on the top, first read the results from the output files created by the executing program, using *FileReader* and *BufferedReader*, conduct post-processing calculation and then send the results directly back to the user, using *response.getWriter*, or create result file in the HTML file using *FileWriter* and *PrintWriter*. The Applet program, linked to the buttons *Graphics1* and *Graphics2* on the top, is in charge of processing the data files and displaying the result curve graphically to the client side, as shown in Figure 6.6. The detail about the applet will be described in the next section.

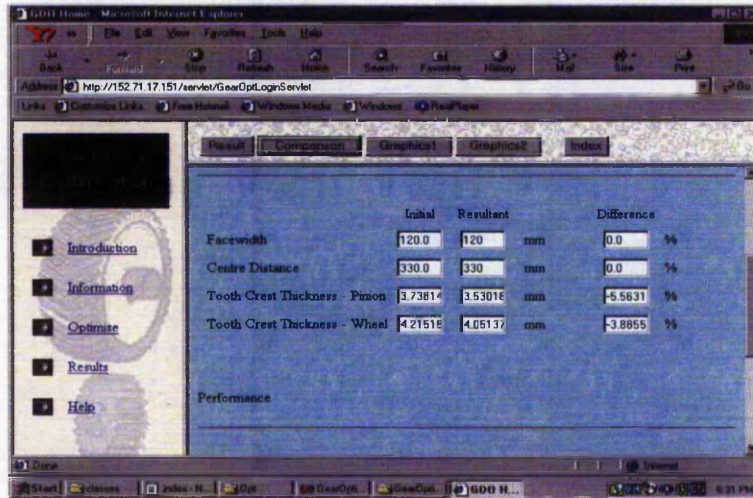


Figure 6.5 Text results shown by the result servlet

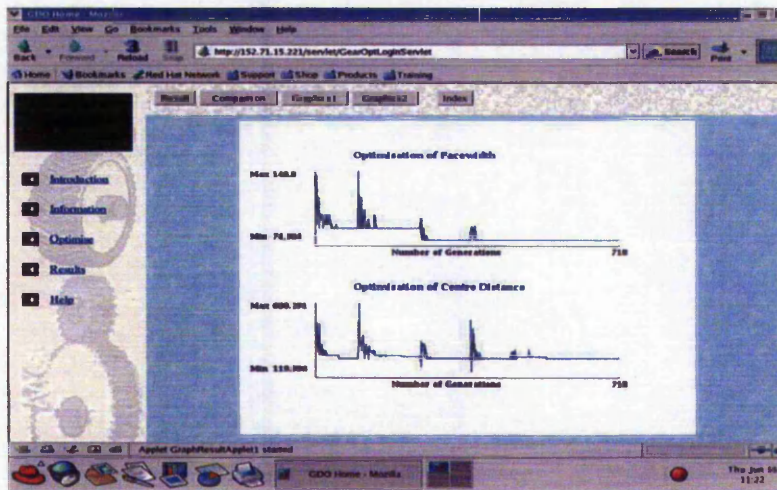


Figure 6.6 Graphical results shown by the result servlet

6.3.3 Applets

6.3.3.1 Features of Applets

An applet is a program written in the Java programming language that can be included in a HTML page. They are defined in HTML language using the <applet> tag. When a Java technology-enabled browser is used to load and view a HTML page that contains an applet, the applet's code is transferred to the user's system along with the HTML page and executed by the browser's Java Virtual Machine (JVM).

Since applets can be downloaded from any site on the World Wide Web and run on a user's system, some security issues have been taken into consideration. Some restrictions have been implemented to prevent malicious applets that contain viruses or Trojan horses, which can cause system damage. The restrictions on applets include the following:

- Applets cannot read or write to the client's file system, which means they cannot delete files or test to see what programs have been installed on the hard drive.
- Applets cannot run any programs on the client's system.
- Applets cannot load programs native to the local platform, including shared libraries such as Dynamic Link Libraries.

However, these restrictions do not limit the process. Applets are needed to communicate with the server and to read the data files in the server. This is permissible, since the files are located on the server where the applets have been downloaded from.

6.3.3.2 Data Retrieval

All the data files that are produced by the optimisation program are saved on the server. A Java applet running on the client's machine needs to access these files either for the progress bar or the graphs. The *URL* class has been used to encapsulate a uniform resource locator. This allows the applet quickly and easily to access the file system on the remote server. The *URL* class specifies the TCP/IP protocol to use either 'http' or 'ftp', the port number (usually 80 for the Web servers), and the exact location of the remote object.

Relative URLs have been used rather than the hard coded URL address to locate the output files on the server. By using the *getDocumentBase()* method of the *URL* class, the file is searched for within the original location of the Web page that contained the applet. This is very useful if for any reason the site needs to be moved, and then the Java code will not need to be recompiled.

6.3.3.3 The Progress Bar Applet

The optimisation programs are mostly computationally expensive. The time taken to perform the optimisation process varies and is dependent upon the population size and the number of times that the process is to be repeated. The results will only be output when the optimisation process is completed. In this situation, if for any reason the connection is lost between the client and the server, it will not cause a problem, since the program is running on the server independent of the client. A flag has been used to check if the optimisation is completed.

In such an environment, it is necessary for the users to know the execution progress of algorithm throughout the whole process. A graphical progress bar is used to achieve this. It is designed to check the progress of the execution continuously and display the progress graphically on a user's screen. Java Applet has the capability of accomplishing the desired task

A status data file is created by the optimisation package on the server. Data within this file indicates the progress of the optimisation in converging on a solution. While the optimisation program is being executed, the data within this file is continuously being updated. The data is utilised along with the number of tests to estimate the time for the process to converge on a solution. They must however be scaled down to the size of the progress bar, before the object is redrawn.

A feature in Java called *thread* has been employed to control the accessing of the data file every second, each time reading and updating data into the progress bar. Anything that runs continuously should run in its own thread. This would help in the reduction of the processing time.

In order to use the thread, the applet must be defined as runnable, by adding "implements Runnable" to the applet class. An instance variable must be defined to hold the applet's thread object. The *start()* method will create a new method and start it running. The actual activities occur in the *run()* method. This is where the data file is read, the necessary calculations to determine the status of the progress bar is done and

the progress bar is repainted. As mentioned earlier, the content of the progress bar is updated every second, i.e. the content of the data file is read every second. To make the activity wait for 1 second before accessing the data file another time, the *Thread.sleep()* method is called within the *run()* method, as shown below:

```
try{ Thread.sleep (1000); }  
catch (InterruptedException e) {}
```

In the *stop()* method, the thread is stopped from executing. The *stop()* method is activated whenever the user leaves the page. If the user returns to the page the *start()* method would restart the thread.

If for any reason, the connection with the server is lost, there would not be any disturbances in the overall functionality of the progress bar. This is due to the fact that the optimisation program is running on the remote server, and the value of the data file that determines the progress of the optimisation is being updated, since it is working independently of the power on the client's machine or the connection between the client and the server. When the connection is established again, the progress bar would jump ahead into the new position.

6.3.3.4 Graphics Applet

When the execution of the optimisation program is completed, in addition to accessing the resultant data, the user has the option of displaying the resultant graph of the optimisation. The performance of the designs is given in the graphs, indicating the trend of the search and the levels of performance. The result traces also provide a means of evaluating the convergence of the genetic algorithm that is indicated by a set of graphics of the objective functions and evolutionary generation. An example of the graph for *Facewidth* and *Centre Distance* is shown in Figure 6.6.

6.3.4 Invoking the Application Object from the Main Servlet

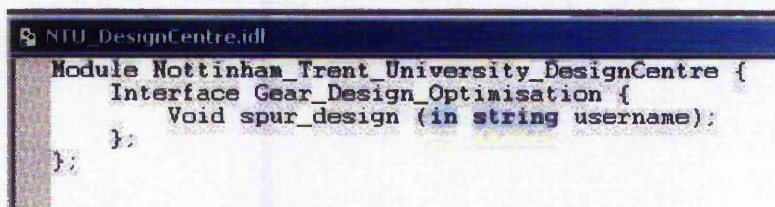
As described in chapter 3 and chapter 4, CORBA is used in this project to establish the communication environment between client programs and application object programs.

In this application paradigm, a servlet on the Web server side is designed as a CORBA client to invoke a CORBA object that is encapsulated from the C++ algorithm program.

6.3.4.1 Defining the IDL Interface

The first step in developing a CORBA service is defining an IDL interface that specifies the type of operations the server will support. In this case, the operation is the whole gear design optimisation procedure.

Figure 6.7 shows the IDL interface for Gear_Design_Optimisation object, on which there is one method (i.e. operation) *spur_design* that has one input string parameter for user path management.



```

NTU_DesignCentre.idl
Module Nottinham_Trent_University_DesignCentre {
    Interface Gear_Design_Optimisation {
        Void spur_design (in string username);
    };
};

```

Figure 6.7 NTU_DesignCentre.idl

6.3.4.2 Compiling the Interface into Java ORB and C++ ORB

In this application, Visibroker for Java and Visibroker for C++ of Borland are used as CORBA developing tools. The compiler *idl_to_Java* is used on Servlet client side to produce Java client ORBs (i.e. stubs) while the compiler *idl_to_cpp* is used on object server side to produce C++ server ORBs (i.e. skeletons).

Server-side ORBs:

- *NTU_DesignCentre_s.hh*: Contains the definitions for the *Gear_Design_OptimisationPOA* servant class.
- *NTU_DesignCentre_s.cpp*: Contains the internal routines used by the server

Client-side ORBs:

- *Gear_Design_OptimisationStub.java*: Stub code for the *Gear_Design_Optimisation* object on the client side.

- *Gear_Design_Optimisation.java*: The *Gear_Design_Optimisation* interface declaration.
- *Gear_Design_OptimisationHelper.java*: Declares the *Gear_Design_OptimisationHelper* class, which defines helpful utility methods.
- *Gear_Design_OptimisationHolder.java*: Declares the *Gear_Design_OptimisationHolder* class, which provides a holder for passing *Gear_Design_Optimisation* objects.
- *Gear_Design_OptimisationOperation.java*: This interface provides declares the method signatures defined in the *Gear_Design_Optimisation* interface in the *NTU_DesignCentre.idl* file.
- *Gear_Design_OptimisationPOA.java*: POA servant code (implementation base code) for the *Gear_Design_Optimisation* object implementation on the server side.
- *Gear_Design_OptimisationPOATie.java*: Class used to implement the *Gear_Design_Optimisation* object on the server side using the tie mechanism.

6.3.4.3 Developing the CORBA Application Server and the Object Implementation

On the server-side there are at least two files to be developed by developers. One is for object implementation, and another is for the CORBA application server. The object implementation file *NTU_DesignCentreImpl.hh* contains all the implementation code of the object, i.e. the legacy code of the algorithm program. The CORBA application server program implements the server class for the server side to invoke the object. It is written in C++ and should do the following tasks:

- Including *NTU_DesignCentreImpl.hh*.
- Initialises the Object Request Broker (ORB).
- Creates a Portable Object Adapter (POA) with the required policies.
- Creates the *spur_design* servant object.
- Activates the servant object.
- Activates the POA manager (and the POA).
- Waits for incoming requests.

6.3.4.4 Developing the Client

Instead of a Java application client, as described in Chapter 4, a Java Servlet is developed to act as a CORBA client. In this application, the main servlet is designed to perform the invocation of CORBA object, in addition to doing common things such as accepting parameters, pre-processing the data, writing data files and returning resultant data to the user as described in section 6.3.2.3. Therefore the servlet client must have the following CORBA-related procedures.

- Initialising the VisiBroker ORB.
- Binding to an *NTU_DesignCentre* object.
- Conductng gear design by invoking method *spur_design* on the *NTU_DesignCentre* object.

6.4 Multi-users Environment

Java servlets are threads that run within a single Java process, which runs alongside the Web server. For the multi-user environment, the Java processes are started and located into the Web server's memory space the first time the Java servlets are requested. It receives all servlet requests from different users and hands each to an appropriate single servlet instance as a thread. Because Java servlets use this multithreaded model within a Java process, there is no need to create a new process for each request and many more requests can be handled.

If the Java process is a simple process without reading the input files and writing the results files, there is no interference between different users in the system described in Section 6.2. The gear optimisation program, however, needs to read input files and to write results to series of output files. Even though the different users are handled by different servlet instances, these input and output files are normally stored in a same default directory. The optimisation program invoked by different servlet instances to serve different users may read from the same input files and write to the same output files. This causes the problem of data file conflict between different users if they are using the system simultaneously, as shown in Figure 6.8.

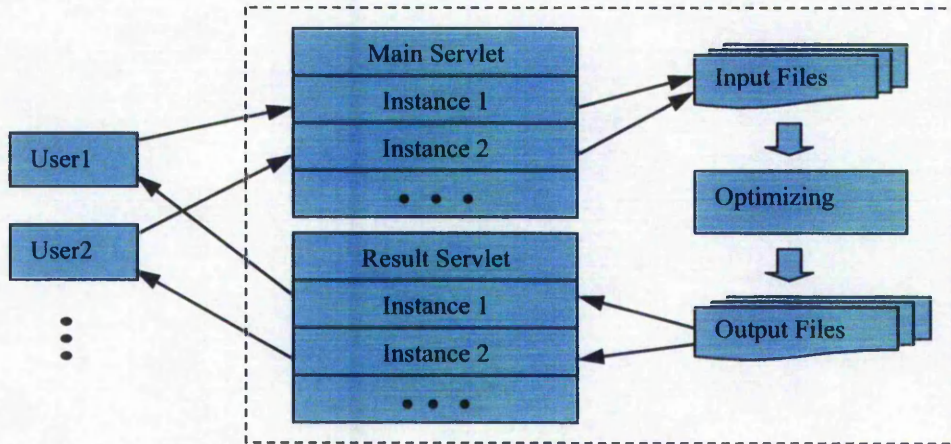


Figure 6.8 Data file conflict between different users

To resolve this data file conflict problem, separate user-specific folders are created for different users on the Web server. Each time the servlet process starts a new instance for a user, it automatically selects the working directory for the gear optimisation program as the user's own folder. When two users, User1 and User2, are using the system simultaneously, all data files for User1 are placed in User1 folder, and for User2 in *User2 folder*. In this way, there would be no data file conflicts between different users, as shown in Figure 6.9.

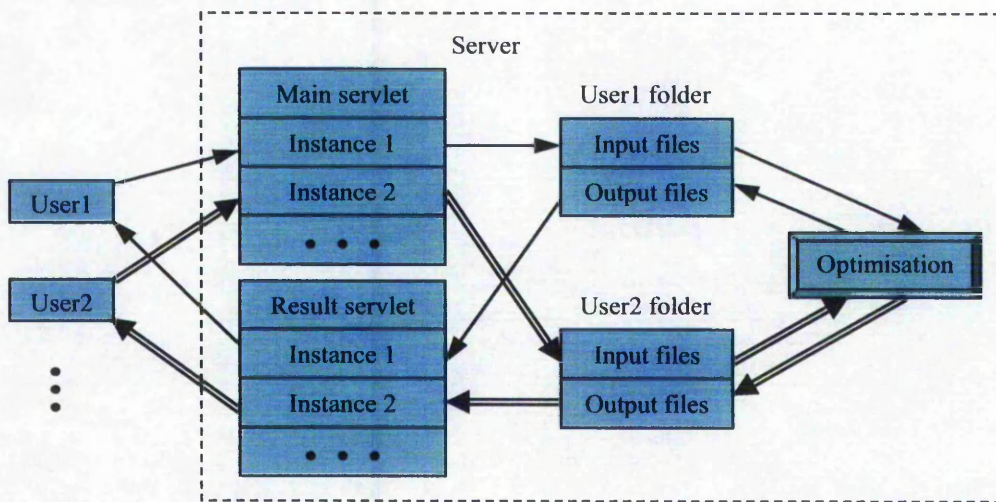


Figure 6.9 Solution for data file conflicts in multi-user situation

The first time the users log in, they are required to input their username and password. If permission is given, then the servlet creates a user-specific folder on the Web server.

This is where all the interactions occur for this user. The input files for and output files from the optimisation program, HTML files and other information relative to the user are placed in this folder. There is no need to copy servlets and optimisation programs to each user-specific folder.

After the optimisation program is completed, the results are written into output files in that folder. So when the user presses the result button, the system will call the output servlet, which reads the correct files in the folder and displays the data results on the returned page. If the user presses the appropriate button resultant graphics will be displayed by an applet, on the page returned by the graphic result servlet. The specific username information is passed by the graphic result servlet to the applet.

6.5 Summary

Java Servlets are small programs that execute on the server side of a Web connection and dynamically extend the functionality of a Web server. Java Applets are small programs that execute on the client side of a Web connection and dynamically extend the functionality of a Web browser. CORBA is used as a mediate mechanism for the communication between a servlet and an existing program.

A Web server-centralised system was developed with combination of Servlets, Applets and CORBA, which enables the existing large design program run remotely on the Internet such that geographically dispersed designers can implement a design on a Web browser and obtain the multiple formats of result from any part of the world.

The utilization of Java Servlets improved the performance in response to the multi-user environment. This study investigated the essential differences between Java Servlets and CGI. The successful application of Java Servlets instead of CGI provides a wider application region. As demonstrated in this chapter, various tasks can be implemented based on Web server by Java Servlets, such as sending different kinds of existing design resources onto the Internet and implementing the remote design of cross-region, communicating commercially with engineering design resources and enlarging the Internet service of a design site, and so on. Multiple-user management mechanism

enables different users utilise the system simultaneously without the data file conflicts and retrieve the latest design results any time.

The combination of Java Servlets and applets and the necessary modification of the gear design program make it possible to master the execution of such a large-sized and time-consuming program, to provide the powerful processing of the results and to display the results graphically to the user.

CORBA aims to provide a communication between heterogeneous applications. In this application, a C++ legacy algorithm application could be invoked by a Java servlet. The servlet program is designed as CORBA client to invoke the CORBA object that is wrapped from the existing C++ program. The connection between servlet client and object is based on same machine and the execution of the program is not affected due to network problem.

Chapter 7 Applet / CORBA Based Worm Gear Design

7.1 Introduction

In engineering there are many existing design programs that are stand alone and can only be executed locally. These programs are ported on the program owner's Web site, which is facilitated with a Web server for providing Internet common services. Design engineer who does not possess design software wants to do the design remotely over the Internet. From the view of a designer, it would be ideal to conduct the design from a common Web browser without any setup work on the client machine. Therefore the designer needs to use a client program with friendly user interface to invoke a service program and further call the design resources.

The Servlet-CORBA system based on Web server-centralised model, described in Chapter 7, is one of approach for this requirement. In this chapter, another approach based on Web server-centralised model is presented, where an Applet-CORBA structure is used to construct the client-server architecture.

A worm gear design program is firstly designed and implemented in C++, and then is wrapped as a legacy program into distributed object based on CORBA infrastructure without rewriting the essential codes. Java Applet can be used as client procedure to invoke the C++ object. Java Applet can be embedded in an HTML page and downloaded in a client machine.

CORBA enables the Applet client to invoke methods on the remote objects at the host server independent of the language that the objects have been written in, and their location. The interaction between client and server is mediated by Object Request

Brokers (ORBs) on both the client and server sides, communicating typically via IIOP (Internet Inter-ORB Protocol).

The Applet client model could benefit from the following Java characteristics:

- Client applets can be downloaded from a Web server and run in any Web browser on any computer worldwide.
- It is not necessary to install clients on users' computers in advance. What is needed is a computer attached to the Internet running any Applet-enabled Web browser or even only an applet viewer.
- There is no need of dedicated client computers for running client-side ORBs (stubs). It provides a stubless client model to keep a thin client.

In this chapter, an applet client based CORBA application for remote worm design is presented, in order to provide a hybrid application sample of CORBA and other WWW technology.

7.2 Architecture of the System

CORBA allows distributed applications to interoperate with each other, regardless of what language/tools are used for the implementation and where these applications reside. In the proposed system, a Java applet is designed as a CORBA client, which communicates with a remote service program written in C++, as shown in Figure 7.1. The communication is transparent. The client program does not need to know the implementation details of the internal object with respect to the object locations (either local or remote). In the system, the client program only needs to know the object's name and how to invoke the object's interface. The ORB takes care of the details of locating the objects, routing the request, and returning the results. The ORBs for a client side is predefined in the Web server side, downloaded along with the client applet and HTML page, and executed in a client machine. Once an invocation session is finished, the memory space in the client machine is released. There is no need to set up for CORBA on a client machine.

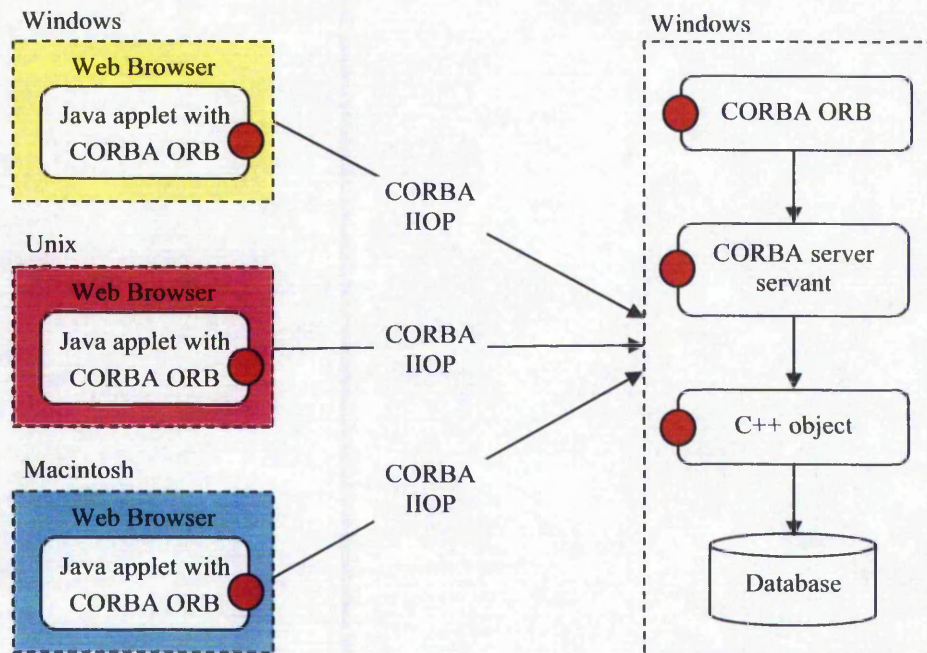


Figure 7.1 Overview of the proposed system

Users of the system can connect a CORBA applet server through browsers. After entering a set of key parameters, browsers automatically download applets from the server to clients. Then the applets run on Java virtual Machine (JVM). Through client's ORBA and server's ORB, clients can connect and access the server, and further invoke the C++ object. The C++ object is the service program, which is connected to the Access database file through ODBC (Open Database Connectivity), a standard database access method developed by SQL.

Clients are Java Applets embedded in browsers, which enables the clients with the independence of platform and security, and so on. Users of the system can use the clients on any platform, such as Linux, at any site.

7.2.1 CORBA IIOP and WWW HTTP

Based on the CORBA IIOP, an underlying protocol, normally a client program, can access remote objects directly without any interference from servers. However Java applet clients cannot directly access the remote object because of Web browser security restrictions placed on Java applets, i.e., the so-called Java sandbox security.

This is not a problem for a Java application client other than an applet client. Also it is not a problem when a Java applet client and server run on the same computer. However this is a problem when a Java applet and CORBA server run on different hosts over the Internet.

On the other hand, in the presence of firewalls, setting up a CORBA IIOP connection probably fails, because most firewalls block every communication, except for some well-known ports like those for HTTP or FTP, which the Web server is based on. Therefore there is a need for a gateway between CORBA IIOP and WWW HTTP to enable this applet client to access remote CORBA objects. Visibroker offers such a gateway named Gatekeeper.

7.2.2 Visibroker Gatekeeper

Visibroker GateKeeper, Borland's product, can be configured to enable Visibroker CORBA applets to communicate with object servers across networks while still conforming to the security restrictions imposed by Web browsers and firewalls. The Gatekeeper serves as a gateway from applet to server objects even if a firewall is restricting access. In case of firewalls, Gatekeeper automatically switches from IIOP protocol to HTTP implementing so-called HTTP tunnelling mechanism.

Without requiring any additional development work, Visibroker Gatekeeper works within the network and security constraints to extend Visibroker CORBA application to the Web and beyond, as shown in Figure 7.2. Clients' requests are firstly sent to Gatekeeper, as IIOP agent, and then gatekeeper transmits the requests to the CORBA object server.

Gatekeeper serves two main purposes: acting as a sandbox proxy and as an HTTP Tunnelling Proxy (for firewalls). As sandbox proxy, Gatekeeper implements the following work:

- Applet communicates with Gatekeeper which then acts as a universal client.
- It takes on responsibility for UDP broadcasting and locating appropriate object servers.

- It receives results and sends them back to client.
- This also allows applets to receive callbacks.

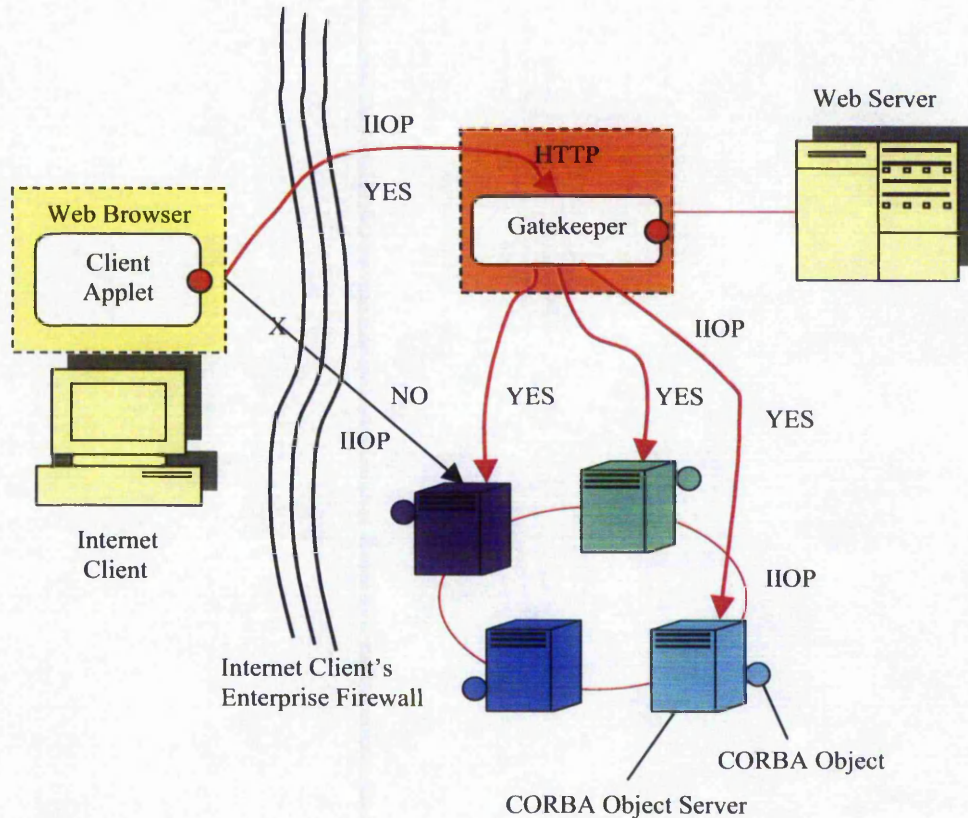


Figure 7.2 CORBA application extends across the firewall with maintaining the integrity and security of the network

As HTTP Tunnelling, Gatekeeper accepts and decodes the HTTP requests. IIOp requests from clients are wrapped in HTTP so that they can get through firewalls. What is more, the Gatekeeper can be used as a Web server.

7.3 Development of the System

7.3.1 IDL Interface Definition

Clients and server use the same IDL interface: `WormDesign.idl`. The IDL file describes the interfaces implemented by the remote objects. Once the IDL file is created, one can use an IDL compiler to generate the client stub code and the server skeleton code. In clients, `idl2java` compiler generates client's stub code by compiling

WormDesign.idl. Clients will use the stub code that implements the methods definition, according to WormDesign.idl, to resolve long-distance CORBA objects. On the server side, idl2cpp compiler generates skeleton code by compiling the same IDL file WormDesign.idl. The skeleton code will be used to transfer clients' service requests. The compiling process of WormDesign.idl is shown in Figure 7.3.

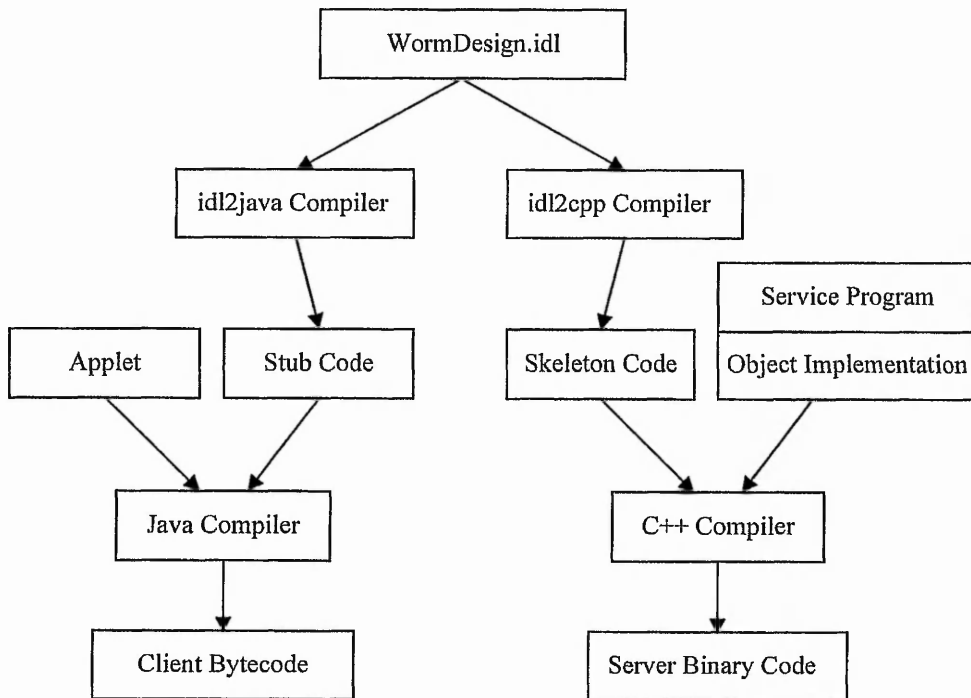


Figure 7.3 The compiling sketch map of client and server interface definition WormDesign.idl

When a client makes a request, the client's ORB locates the object implementation, activates the object if necessary, delivers the request to the object service program and further to the object, and returns the response to the client. The client is unaware if the object is on the same machine or across a network. A client program uses a remote object by obtaining a reference to the object. Object references are usually obtained using a Naming service. Then the object's reference will be passed to the stub code. With the help of the stub code and ORB, the client can transfer the request to the skeleton code and service program, then to the target service object.

7.3.2 Applet Client

As a CORBA-enabled client, the applet binds to the *WormDesign* object. In order to do this, ORB initialisation must be done in the *init()* method of the applet. A client user could run the applet to pass the input parameters, do the design calculation method, obtain the resultant data and view the automatically generated 2D drawing, through invoking the remote methods such as *getData()*, *GADesign()*, *returnData()* on the remote object *WormDesign* and the *2D_Drawing()* method on the applet.

In the HTML Web page that embeds the applet, as shown below, the parameter *org.omg.CORBA.ORBClass* within the *<applet>* tag causes Visibroker ORB to be downloaded. The value of parameter *vbroker.orb.gatekeeper.ior* points to Gatekeeper IOR (e.g. *http://gatekeeper_host:8088/gatekeeper.ior*).

```
<applet code="WormDesign.class" width="200" height="80">
  <param name="org.omg.CORBA.ORBClass"
    value="com.inprise.vbroker.orb.ORB">
  <param name="vbroker.orb.alwaysTunnel" value="true">
  <param name="vbroker.orb.gatekeeper.ior"
    value="http://152.71.15.221:8088/gatekeeper.ior ">
</applet>
```

7.3.3 Object Server

The server side mainly includes the entry of the server side ORB, a server main program, the implementation of objects' methods, and the activation of an ODBC driver. The server side ORB, i.e. skeleton program *WormDesign_s.cc* and its head file *WormDesign_s.hh* are generated automatically by the mapping tool *idl2cpp*. The client stub *WormDesign_c.cc* and its head file *WormDesign_c.hh* are also generated by *idl2cpp*, as shown in Figure 7.4. In this application the client C++ procedures are replaced by Java procedures: Applet.

The server program *WormDesign_Server.C* file is normally developed by the server-side programmer. The server implementation includes the procedures such as initialising the ORB, creating a persistent POA named *worm_agent_poa*, creating an

instance of the *wormdesign* servant, activating that servant on the *worm_agent_poa* and then starting and waiting for client requests.

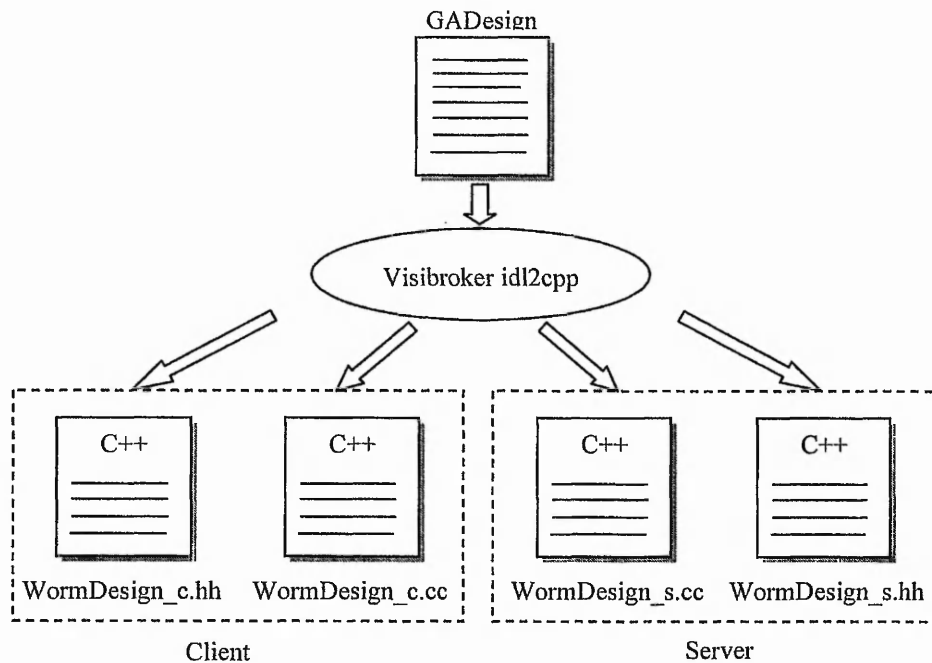


Figure 7.4 The C++ files generated by the idl-to-C++ compiler.

7.3.4 Object Implementation

The *WormDesign.C* is the object implementing programs. The method codes on the object are put in these programs. The method *getData()* is designed accept the input parameters from the user. The *GAWormOpt()* is a Genetic Algorithm procedure for minimised design of stress and for recoding the output data into a database file and the *returnData()* is for returning the resultant data in the database file to the client side.

7.4 Deploying and Running the Programs

The following are the steps to deploy and run the programs:

1. Compile the IDL and the generated Java classes in the usual way.
2. Place the applet, the html file and the ORB class files (both client-side and server-side classes) to a directory on the Web server.
3. Start servlet container Tomcat.
4. Start the Visibroker Smart Agent (OSAGENT).

5. Start the server application.
6. Start the Gatekeeper.
7. Point a client browser to the URL for the application HTML file.

Gatekeeper provides a combined HTTP and IIOP server stack; this allows the applets to interact with object implementations on hosts other than the Web server. It also allows IIOP messages to be transported through firewalls disguised as HTTP packets (HTTP tunnelling). When starting the gatekeeper, please be aware of:

- Gatekeeper should be started in the same directory as the server application
- If gatekeeper is being used as a Web server, then the URL should contain the port used by gatekeeper.

A user could do the design from the Applet GUI in the HTML page, as shown in Figure 7.5. Firstly, necessary parameters need to be input on the left area. The input parameters include the upper limits and lower limits of the design space, and user name input. By pressing “*start design*” button on the right area, the resultant data will be displayed on the right data area. A 2D worm gearing drawing will be automatically created and displayed on the middle area by pressing the *drawing* button on the right area. The *erase* button helps to clear the drawing area. Parameter modifications will yield different resultant data and different sizes of drawings.

The session between the applet client and remote worm gear design object keeps until deliberately stopping the page in this Applet-CORBA model. Therefore the user could easily do multiple trial designs without interrupting the session, while the session is ended with a returning dynamic page on the Servlet model.

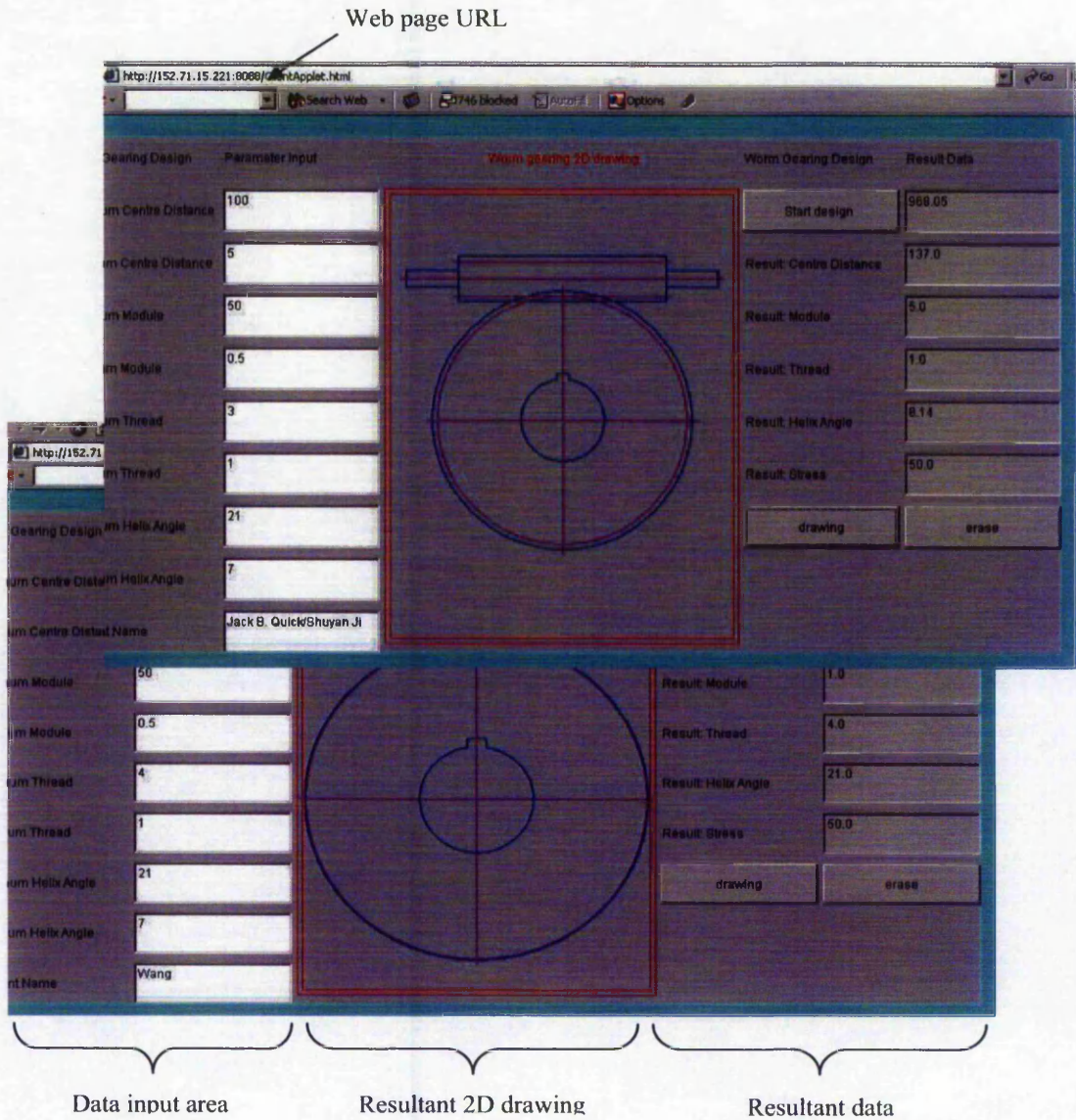


Figure 7.5 The GUI of the worm gear design application from the Web browser

7.5 Summary

In this chapter, Applet-CORBA combination provides a thin client model to enable a user do the remote invocation from a Web browser. As a user of this application, a designer does not need to know anything about CORBA knowledge. The rich features of applet make client developers to design various forms of Web browser application, such as dynamic data, dynamic drawing and picture, animation, and so on.

CORBA itself relies on the IIOP binary protocol to establish the communication between the client and the server. Applet client is not allowed to deal with the coming connection due to the security problem such as Applet sandbox and firewall. In order to fulfil this type of application, Visibroker Gatekeeper is used to establish an IIOP-HTTP bridge by which the CORBA applet acts as a universal applet to deal with the IIOP message wrapped in HTTP. Therefore the barriers of applet sandbox and firewall could be got through.

In this Applet-CORBA based system, the session between a client and an object is not interrupted until deliberately exiting, and this facilitates users to conveniently use the system. How to deploy and run the programs has been demonstrated at the end of this chapter, taking the worm gear design as an example.

Chapter 8 Results and Discussions

8.1 Introduction

The increasing complexity of modern products, the cruel competition pressure and the globalisation of product development necessitate a collaborative design environment where different computer programs and distributed experts in similar or different domains need to be collaboratively involved on a common design activity, in order to obtain high quality and low cost product design. This project aims to construct a Web-based system at lower cost, to support collaborative design over the Internet and thereby shorten the product development lead-time.

From the investigation and review of current collaborative systems and current implementation technologies, it can be found that the latest Web technologies could be used to establish an open architecture based on the existing Web infrastructure for communication to facilitate collaborative design activities. Most of the current systems are still under proof-of-the-concept prototype development stage. Even though some CAD/CAM packages, such as Windchill for ProEngineer, enable the users to collaborate over the Internet, the Internet-based collaboration can only be operated with the same software in the same type of platform and operating system. It is clear that challenges in these areas will remain as a research opportunity.

Dispersed design applications may use different programming languages, such as C/C++, Python, Java, Perl, or Cobol, and run in different computing platforms such as UNIX, Microsoft Windows, IBM OS/2, or Apple Macintosh. This needs to be taken into account to enable these heterogeneous applications to be integrated at lower cost and work together smoothly.

System development cost is another inevitable problem to be taken into consideration. Reuse of legacy systems is a methodology for reducing the cost of software development and maintenance. It is necessary to build an environment to accommodate many existing design and manufacturing applications without rewriting the essential codes.

Gear design is a typical topic in engineering design area. A gear design, especially addendum modification design of spur and helical gears involves multiple design objective evaluation and multiple constraints check such as gearing intervene check, undercutting check, and so on. Several systems have been developed and implemented to increase design productivity in the gear industry [123-128]. Most are standalone and not designed based on the distributed structure.

For a complex product design such as a gear design, there may be a need for invocation of multiple programs during a complex design procedure. These applications should be invoked dynamically according to the user requirement.

In a collaborative environment, affections to a design from multiple design domains should be leveraged to retrieve rational or optimal design solution. A proper mechanism needs to be designed to help designers to balance multiple design interests.

In the engineering field, there are many large-sized computing applications for product design and analysis. These programs are not convenient to be downloaded on the thin client machine and likely run on the owner's previous machine. System design needs to facilitate the parameters input, program execution and monitoring, and result viewing.

The solution lies in the combination of distributed object technology, Internet-oriented language Java and Genetic Algorithms. A collaborative environment, which integrates a combination of these technologies with engineering design to help diverse designers to conduct design over the Internet have been developed. In the developed system, heterogeneous and distributed design applications can be wrapped into Web-enabled

component to be invoked remotely. Based on the Web-based architecture, not only can multiple users share an identical design application or design data, but also an active designer might use multiple design resources simultaneously. The resultant Web-based architectures have identified and addressed three main features: collaborative distributed design optimisation, remote execution and monitoring of singular large-size and time-consumed programs, and session-oriented on-line design with parametric drawing.

8.2 Development of the Web-based Architecture

It is necessary to investigate the distributed technologies to develop a Web-based system. In Section 3.3.2, CORBA and other Web technologies are compared. RMI addresses the remote method invocation across platforms but only between Java applications. DCOM is the distributed component object model for Windows and thus not for platform-independent circumstances.

In this research the heterogeneity is one of the main factors to be taken into account to construct a collaborative environment for integrated gear design. Therefore CORBA is thus utilised to develop a distributed collaborative environment, where it is possible to invoke heterogeneous distributed design applications, which might be written in different languages or running in different platforms, to enable collaborative design over the Internet.

An emerging distributed middleware technology Web Services technique, in addition to CORBA, is also investigated. Even though it can be applied for almost all the situations CORBA is used for, Web Services aims to obtain more flexible and more common features because using XML to send data through HTTP protocol. However, another of the main factors to be taken into account in this research is efficiency. CORBA IIOP communication protocol ensures the communication efficiency because its binary mechanism.

Integration and communication of different applications are presented in Section 4.7 and 4.8 (C++ on Linux and C++ on Windows), Chapter 7 (Java Applet and C++),

Chapter 6 (Java Servlet and C++). All the aspects relevant to the integration and communication of these applications are revealed, including the integration architecture, the interface implementation, the underlying communication mechanism, the programming issues on client side and server side, and case study (in Chapter 4, Chapter 6, and Chapter 7).

In addition, legacy system reusability, openness and scalability are necessary elements to be considered. Using CORBA, it is easy to implement these features because of its IDL the interface definition.

The wrapping structure and developing procedures of legacy programs are described in Chapter 4. The wrapped legacy object based on CORBA could work together with new CORBA-ware applications as all the distributed objects have the identical interface definition, i.e. IDL. Therefore the system is open and scalable. New object is added in dynamically and worked together with previous ones without changing original system.

This research also investigates the distributed system architectures and explores the possibilities in supporting collaborative design. There are two kinds of system architectures for distributed collaborative systems, i.e. the centralised model and point-to-point structure.

In the centralised model, it is easier to facilitate design services such as administration, registration, searching, resource storage, etc. Collocated designers could download design information or design model from the server and edit the information or model using local design programs. The collocated designers could also invoke the applications on the central server side using proper mechanisms instead of downloading the applications onto the client machine. The key advantage of the centralised model is that it is easier to ensure data consistency, as there is only one master copy saved in the central database. However, all the activities are conducted through the central station and thus it leads to a central bottleneck. Implementing collaborative work using this model typically requires high network bandwidth.

In the point-to-point model clients could contact applications owned by partners without going through a central station and thus it would avoid a central bottleneck. This leads to an effective communication structure. Each node has equivalent capabilities as well as responsibilities that either request or provide service, differing from the centralised model where the central database is dedicated to serving the others. The distributed applications could be easily invocated synchronically to implement a common task according to a proper procedure. Therefore collaborative work on this model does not need high network bandwidth.

In order to develop a powerful and robust system with high efficiency and easy-to-managing features to support different design circumstances, in this research, not only two basic structures but also hybrid architecture of these two structures are used to construct collaborative design environment. The systems based on Web server structure are developed and described in Chapter 6 and Chapter 7 while the point-to-point communication mechanism and convenient Web server-centralised management are combined to be used in the distributed gear design systems, as seen in Chapter 4.

Java, an Internet-oriented language, is considered to combine with CORBA in this research. Java provides the client side GUI developing tools and server-centralised server extension tools to enhance the platform-dependent functions in distributed systems. Java applets are not only capable of simple interaction with the user, but also capable of taking part in complex interactions with backend services in CORBA-based distributed system. A combination of Java with CORBA for application integration presents a good solution for application components capable of accessing multiple backend services distributed across the Internet. CORBA and Java are complementary to ensure high source portability and robust platforms for Web-based applications. All these features are revealed in Chapter 6 and Chapter 7.

8.3 Paradigm 1: Distributed Gear Design Optimisation Using Web Technology and Genetic Algorithms

Gear design involves multiple considerations from many domain experts. Collaborative engineering enables multiple specific domain experts and their computerised tools to be integrated to work together for a common design task.

From the studies [123-128], it is not difficult to find that there is still short of a Web-based, fully-integrated, distributed spur gear design system that comprises all the necessary components for integrated gear design, including user requirement specification, design optimisation, stress analysis, 2D/3D modelling, graphical file output, and so on. In order to support the cooperative design over the world based on the Internet, there is a need to make all the gear design resources including gear design experts, their design tools, and databases work collaboratively together in a platform over the Internet.

A Web-based infrastructure is explored and developed using CORBA, where all the heterogeneous design applications can be recognised and invoked remotely. Based on the architecture, gear design applications, for stress calculation programs, undercutting check, gear interference, etc, can be wrapped into Web-enabled objects to be accessed.

Gear design, like most design problems, is difficult and complex, and affections of the models from different specialists to the design solution are comprehensive. Therefore, in the collaborative design environment, a resolving strategy to such a complex situation is inevitable for an active designer (or main designer) to leverage multiple interests. In this research, gear design is treated as a design optimisation model. Considerations from different domains are thought as design objectives or design constraints. Genetic Algorithms (GA)-based mechanism is developed to help users (designers) to conduct gear design optimisation based on the distributed system.

In the development for the distributed gear design optimisation system, the following achievements are obtained.

- The architecture of the system

- Heterogeneous application communication implementation
- Unified Graphical User Interface (GUI) design
- Optimiser implementation
 - Cascaded GA algorithms design
 - Bit field structure and union for encoding
 - Variable dimension problem
 - Variable penalty function using Simulating Annealing
 - Multiple objectives optimisation
- Automatic 2D gearing gears profiles generation tool

8.3.1 The Architecture of the System

To develop the Web-based system for gear design optimisation, it is necessary to consider basic requirements from the views of the system users, i.e. designers, program providers and developers (Section 4.3.1). Designers wish to use remote resources as much as required without enlarging their own systems. The developers like to use their own favourite languages to develop relevant interface. Program owners want the design application to run on previous machines and previous operating systems. The system developed using CORBA can meet all the requirements. All the gear design application programs, written in C++ and Java and run on Windows and Linux, are wrapped into distributed components to be used remotely by other experts using IDL. Through ORB and IIOP communication mechanism, these applications can be communicated with each other. The system structure and its working mechanism are described in Sections 4.3, 4.4. As a distributed system it has the following functions:

- **A unified graphical user interface (GUI)** -- A user of the system merely interacts with the unified graphical user interface (GUI) to conduct all the design activities and does not need to know any technical details about communications between the interface and remote resources. The unified GUI provides an integrated design environment that has design variables and design objectives set-up, application circumstance identification, monitoring design optimisation process, viewing and analysing multiple solution sets, gearing movement simulation, and graphical file output.

- **An optimiser helping to conduct multiple objective design optimisations --**
An optimiser is included in the GUI program and designed to help users to conduct multiple objective design optimisation in the collaborative environment. The optimiser procedure is designed based on Genetic Algorithm. To each solution of design variables, evaluation according to multiple design objectives and design constraints needs to be done through invoking relevant objective calculation and constraint evaluation programs, which might be located remotely. By using the optimising procedure the optimiser provides, the user could implement the gear design optimisation among multiple design interests from different design experts.
- **Design data visualisation in graphics --** Through the integrated design environment a user (designer) can view design results both in text data and graphics. A 2D gear profile can be generated automatically according to the design data.
- **A designer could conduct integrated gear design at limited time –** A designer could conduct gear design to find the rational tuning parameters in consideration of multiple aspects in design or manufacture, within limited time without augmenting his own system. Therefore the distributed collaborative design system enables the development cycle of product reduced.
- **Supporting multiple user environment –** All the objects are implemented in multiple-thread model and thus multiple clients could synchronically share the same objects.
- **System openness –** Any new application components can be integrated together with previous objects, recognised and invoked as long as they are wrapped into CORBA objects.

8.3.2 Heterogeneous Application Communication

Implementation

In the presented distributed gear design optimisation system, Windows and Linux platforms, Java and C/C++ languages are used for exploring the possibility of platform-independent and language-independent features.

In the engineering field, there are some computing programs ported on the Linux operating system. Windows are a popular operating system for common users. In this research, communication between C++ stress computing programs on Linux and VC++ GUI program with an optimiser on Windows is implemented. The developed communication mechanism enables C++ program on Windows to invoke C++ on Linux without correcting the essential codes. Programs on Linux are wrapped as CORBA objects and still ported in previous operating system and debugged in previous developing tools GCC. VC++ program on popular Windows can be designed as CORBA clients to invoke C++ program on Linux.

The achievements of this research can be summarised as follows:

- **Selection of proper tool** – There are many CORBA developing tool for implementing the communication between two heterogeneous applications. Different tools are used for different circumstances. For example, OmniORB is used for C++ and Python applications on Linux and Windows. Visibroker for C++ and Visibroker for Java are used to develop the communication procedure between Java and C++ on Windows and Unix. To Linux, there is another Visibroker package to implement it. Since the communication speed between the optimiser and remote objects is crucial to the whole system OmniORB is used as the developing tool because it is the fastest available C++ ORB and completely free. It can be used to implement the communication between C++ on Windows and C++ on Windows, C++ on Linux and C++ on Linux, and C++ on Linux and C++ on Windows (Chapter 4).
- **Objects interface definition** – CORBA IDL is used to define the object interface, only including object name, method name, and input and output parameters. The interface program can be mapped into client ORBs and server ORBs, which is in charge of communication (Chapter 4).
- **Development on Linux server side** – On the server side, object implementation and server program are written. The object implementation is for gears contact and bending stress computing. Server program includes ORB initialisation and bounding to the object. Server program, object

implementation and the automatically generated ORB procedures are compiled and built using C++ tool GCC on Linux, as described in Section 4.7.

- **Development on Windows client side (Chapter 4)**– Developers on client side need to write C++ client program on Windows. The optimiser procedure is designed as the CORBA client, included in a GUI program. The optimiser program contains the ORB initialisation routines and referencing mechanisms for the invocation of the remote objects. There are three approaches to integrate and debut the VC++ optimiser program pieces and CORBA routines. **a)** The optimiser is separately developed and included in CORBA routines as a process; **b)** One program combining CORBA routines with the optimiser procedure is debugged in VC++ environment with CORBA-related setting up; and **c)** One program combining the optimiser procedure with the CORBA routines is debugged in command line makefile of CORBA. The above b) and c) provide a higher running efficiency model than the process model provided in a).

8.3.3 Unified Graphical User Interface (GUI) Design

As a user of the gear design system, a designer wishes to conduct integrated design in a visual environment. As described in Section 4.6, the system provides a unified graphical user interface, written in VC++, with the following functions:

- **Design objectives identification** – There are six design objectives to be selected in the gear design system. Each objective can be appointed by a weighting coefficient according to design intentions. Each of the objectives is associated with a local or remote evaluation program.
- **Design variables configuration** – There are up to 9 optimisation variables defined in the system. The system supports the flexible design for the less than 9 or up to 9 of the variables, e.g. dimension variation. The un-configured variables need to be appointed by a given value.
- **Specify other design requirements** – In addition to the design objectives and design variables, users need to specify other design requirements, including application parameters, quality parameters and material parameters.

- **On-line help** – During the input process, pressing F1 button, on-line help dialog can be activated to provide the relative knowledge information about the current activities. The on-line help is developed using Microsoft Visual Studio tool, which is involved into the VC++ developing project.
- **Input data selection** – There are many design parameters to be entered. To facilitate users to input these parameters correctly and quickly, the application program provides four flexible data input selections, including user default design parameters, system default design parameters, new design parameters, and retrieving existing design parameters.
- **Calling optimiser program** – The optimiser program is implemented separately from the GUI procedure. Pressing the “start” button, the optimiser program can be invoked.
- **Viewing design results** – After finishing the optimisation procedure, pressing “result” can display the data results and graphical results.

8.3.4 Optimiser implementation

In the distributed design system, an optimiser is developed as a design making aid to help the designer to do multiple objective design optimisations over the Internet. The optimiser is designed based on Genetic Algorithms (GAs). Gear design optimisation involves the high number of variables, objectives, and constraints. Variables take continuing and discrete values. Genetic Algorithms (GAs) is utilised as the optimising approach to fulfil the design optimisation. GAs allow spatial relations to be taken into account without the requirement of linearity or continuity of the evaluation function. They use only objective function information, not derivatives or other auxiliary knowledge. GAs search from one population of solutions to another, rather than from individual to individual.

In the development of the optimiser, this research has received the following achievements:

- Gear design optimisation model
- Cascaded GA algorithms design
- C++ bits-field and union structure for chromosome encoding

- Variable dimensional problem
- Variable penalty function using Simulating Annealing
- Multiple objectives optimisation

8.3.4.1 Gear Design Optimisation Model

Gear design is one of the classical topics of mechanical engineering design. The classical route followed for the design of gears is to appeal to standards, such as BS AGMA, DIN or ISO [92-98]. These standards are based on extremely large collections of results and empirical rules from practical experience in a vast range of engineering applications. They provide a set of formulae, rules and charts to design the gearing taking into account various working conditions and several aspects of their performance, such as the power level, noise, lubrication conditions, wear rate, likelihood of impact, pitting, and corrosion. In this project, addendum modification design problem of spur and helical gear design is used as case study to explore the possibility of the proposed collaborative system for improving design efficiency, as presented in Section 5.2.

The addendum modification design problem of spur and helical gears is modelled in up to 9 design variables, up to 6 design objectives and 24 design constraints. The design variables includes the geometrical information such as module, face width coefficient, pressure angle, helical angle and pinion tooth number; addendum modification information; and manufacturing tool information such as addendum coefficient and rack tip radius coefficient. The design objectives involve spatial considerations including minimising face width and minimising the centre distance, performance considerations including reducing the bending stress, contact stress, the difference in bending stresses between the pinion and wheel teeth, and the difference in the tooth tip sliding ratio of the pinion and wheel. The constraints includes the considerations of bending and contact strength, cutting interference condition, tooth tip thickness, interference at the roots of mating gear teeth, slide/roll ratio for the gear tooth tip that is designed to evaluate the lubricating feature, rack tip fillet radius coefficient limit, contact ration, and so on.

8.3.4.2 Cascaded GA Algorithm Design

Calculation efficiency of GA Algorithm is one of the most important factors to be taken into consideration. Cascaded genetic algorithm structure is designed to gain high efficiency of coarser search in global space and high accuracy of finer search in smaller space, as presented in Section 5.3.2. The solution gained from the first tier is used as the warm value set at the second tier. The smaller space around the warm value set is used as the search space at the second tier.

8.3.4.3 C++ Bits-field Structure for Chromosome Encoding

In this gear design optimisation program, population ranges from 100 to 10000. Generation sometimes need to reach a very big number, e.g. 10000 or even more. During the evolution process, every chromosome in the population must be the encoded and decoded, which means a huge amount of computation is involved. Therefore, the program design of chromosome data structure has a vital effect to the execution efficiency of the program.

C++ Bit field structure in conjunction with union data type is used to provide an advanced data structure for chromosome representation, in order to save computer memory units and improve computing efficiency.

A union is a memory location that is shared by two or more different variables, generally of different types. The union structure is used for the bit field variable sharing the same memory with the unsigned long integer variable. The bit field represents binary code for a chromosome while the long integer variable represents the encoding value for each gene in the chromosome. This structure using bit field and union provides a convenient way to retrieve the encoding value from every gene saving the calculation from the binary expression to encoding value, consequently, so that the running efficiency of the program is greatly improved. Section 5.3.3 illustrates the detail in bit-field design.

8.3.4.4 Variable Dimensional Problems

In this research, the gear design optimisation is defined as up to 9 variables. However, in practical applications, it is not always necessary to select all of the 9 variables at the same time; some of variables are sometimes given by a certain value and not variables any more. The dimension of the application has been changed.

Normally, in the representation for a chromosome, a fixed binary string represents the fixed number of variables. The corresponding genes of the unselected variables are involved in the evolution, and their values are still kept in the given values in decoding, instead of being mapped from the evolutionary operations. That may causes redundant gene segments and mapping deceiving.

In order to resolve this redundant gene segments and mapping deceiving problem, in this study a new concept of dynamic and variable length chromosome is devised, as described in Section 5.3.4. The dynamic chromosome consists of the gene segments of the selected design variables. For those unselected design variables, their corresponding gene segments will not be included in the dynamic chromosome and, consequently, these gene segments will not be involved in the evolution process. In this way, the redundant gene segments and mapping deceiving problem can be overcome. Dynamic mapping array is created to obtain the dynamic combination of genes according to user's variable dimension requirements.

8.3.4.5 Variable Penalty Function Using Simulating Annealing

The gear design optimisation involves huge number of design constraints. Penalty function to constraint violence dramatically affects the process of convergence and the quality of the result. If a set of penalties is too harsh, then few solutions that do not violate constraints will quickly dominate the mating pool and yield sub-optimal solutions. A penalty that is too lenient can allow infeasible solutions to flourish as they can have higher fitness values than feasible solutions. Often, the algorithm must be rerun many times before a combination of penalties is found that allows infeasible solutions to die and feasible solutions to flourish. The main difficulty in applying penalty functions is that they are problem dependent.

Variable penalty function based on simulating annealing algorithm is constructed to deal with the constraints, which provides another approach to penalise the constraint violence. The variable penalty function contains an attenuation factor that is related to the constraints and the temperature schedule in the simulating annealing algorithm. The attenuation factor makes the penalty to the constraint violence from gentle to harsh over the convergence process, so that some infeasible individuals can be kept in the population at the early stage of the evolutionary process to avoid the process to stop prematurely. The construction of the variable penalty function is described in Section 5.5.1-5.5.3.

Many instances of optimisation utilising variable penalty approach are illustrated and the results of these instances are used to verify the performance of variable penalty approach and analysis the influence of the temperature schedule and the initial temperature parameter (Section 5.5.4).

8.3.4.6 Multiple Objective Optimisation

Multiple-objective optimisation is investigated and the weighted sum solution is given. Changing the weighting factors is employed to obtain the entire collection of solutions for user to select final solution from wider region. The methods of choosing appropriate weighting factors are discussed and, the gear optimisation results obtained by applying these methods are illustrated.

The weighted sum solution provides a simple but effective approach to obtain rational design solution collection stead of an optimal solution, which is actually impossible to obtain when multiple controversial objectives are involved in the problem. It is proved that each solution by using this approach is Pareto solution.

In the distributed gear design system, multiple objective optimisation provides a underlying technique to design the optimiser as a design making aid, in order to help designer to leverage the interests from different domain experts.

8.3.5 Automatic 2D Mating Gears Profiles Generation

In the development of distributed gear design optimisation, a 2D gearing gears profiles generation tool is developed. To each solution of design optimisation, a corresponding 2D can be produced by the generation tool according to the resultant data. The involute profiles of the gearing gears can be precisely drawn and displayed (Section 4.5.1). Meanwhile, by using the generation tool a corresponding DXF file can be also output. The DXF file, as an exchange graphics file, can be displayed in popular CAD software. This tool is written in VC++. The details are given in Section 4.5.

8.4 Paradigm 2: Remote Invocation of Singular Large-scale Computing Program Using Servlet and CORBA

The distributed gear design optimisation provides an integrated model with multiple programs. In engineering, there are many computing programs, which are large-scale and time-consuming, and then not convenient in multiple programs working environment. Therefore it is necessary to design a communication mechanism for the remote invocation of singular large-scale computing program.

There is legacy design application package, which includes a GUI written in VB and an algorithm program written in C++. To each calculation, the package produces input and output files. The algorithm program is large scale and time-consuming. This project aims to make this existing package Web-enabled to execute remotely over the Internet. Chapter 6 describes the details on the development of this system.

A Web server-centralised system was developed with combination of Servlets, Applets and CORBA, which enables the existing large design program run remotely on the Internet such that geographically dispersed designers can implement a design on a Web browser and obtain the multiple formats of result from any part of the world.

A user, i.e. a design engineer could conduct the gear design including inputting parameter, activating calculation program, monitoring the progress of the program execution, and viewing resultant data and data analysis, within the Web browser, just as visiting common pages.

Java Servlets, small programs that could execute on the server side of a Web connection, are developed to dynamically extend the functionality of a Web server. The developed Servlet programs could retrieve the data passed in the page, parsing them and further passes to the execution program when they activate the program. Java Applets, small programs that execute on the client side of a Web connection, are developed to dynamically extend the functionality of a Web browser. The developed applets that are included in the response page returned by Servlets could implement the display of resultant text data, resultant graphics, and program execution progress bar.

CORBA is used to provide a communication between the Java Servlet program and the large-scale programs on the server side. In this application, a C++ legacy algorithm application could be invoked by a Java servlet. The servlet program is designed as CORBA client to invoke the CORBA object that is wrapped from the existing C++ program. The connection between servlet client and object is based on the same machine and the execution of the program is not affected due to network problem.

The session Servlets provides, between two computers, could not stay but stop with the response page by the servlet. The model Applet/CORBA provides is session oriented. The Applet/CORBA session between two computers could be kept unless the connection is deliberately stopped, which is used for the development of on-line worm design system.

8.5 Paradigm 3: On-line Worm Design Using Applet/CORBA

By using CORBA and Java applet, a plug-in running model of remote program for worm gear design is given. A session established between point-to-point can be kept

linking until stopped manually. This model is used for the flexible need for parameter modification repetitively. The development of this system is described in Chapter 7.

Applet-CORBA combination provides a thin client model to make user do the remote invocation from a Web browser. As a user of this application, a designer does not need to know anything about CORBA knowledge. The rich features of applet make client developers to design varieties forms of Web browser application, such as dynamic data, dynamic drawing and picture, animation, and so on.

CORBA itself rely on the IIOP binary protocol to establish the communication between the client and the serve. Applet client is not allowed to deal with the coming connection due to the security problem such as applet sandbox and firewall. In order to fulfil this type of application, Visibroker Gatekeeper is used to establish an IIOP-HTTP bridge by which the CORBA applet acts as a universal applet to deal with the IIOP message wrapped in HTTP. Therefore the barriers of applet sandbox and firewall could be got through.

An applet could be linked to multiple CORBA objects with the same ORB. This provides another integration model of multiple programs, without any setting up for CORBA on the client machine.

Collaborative product design may involve a number of software applications that run on geographically distributed computers. For example, designers, material specialists, manufacturing engineers, and structural analyses of a product may reside in different locations and use separate computer systems and software packages for design and analysis.

Chapter 9 Conclusions and Future Work

9.1 Conclusions

In this section the conclusions have been based upon the aims and objectives of the research as outlined in Section 1.3. The Web-based technologies and Genetic Algorithms have been applied to develop the collaborative environment for integrated design. The following specific conclusions can be drawn from this work.

1. CORBA was found to be appropriate to construct a distributed infrastructure to develop Web-based systems for integrated design. The developed Web-based infrastructure based on CORBA facilities with many necessary features:
 - **Heterogeneous application integration and communication** – Based on the CORBA-based system, many different applications, in engineering area, that might be ported in different computers, running on platforms and different operating systems, and written in different languages, can be integrated together and interoperated by each other. Using correct CORBA developing tools these heterogeneous applications can be encapsulated with a unified interface and then recognised and invoked with each other. The deployed objects can be integrated in flexible model for different application circumstances and therefore it is possible to develop distributed application systems for collaborative design over the Internet. For example, multiple designers could share a remote application (in Chapter 6, Chapter 7) through client programs or a main designer could use multiple remote applications for complex and integrated design through a unified user interface (Chapter 4).
 - **Reusing legacy applications** – In engineering area, there are many applications that are traditional, stand-alone, single-user computer-aided applications that are still valuable to perform crucial work, and usually

represent a significant investment and years of accumulated experience and knowledge. It is found that functionalities of these legacy applications can be extended by employing modern distributed object computing technology CORBA. CORBA IDL is used to define the unified interface for all the applications including legacy applications regardless of their languages and platforms. Without rewriting the essential codes, a legacy application can be wrapped into CORBA object implementation. The wrapping structure, interface definition, and combining model and debugging approaches regard to the essential codes with communicating procedure, are demonstrated (in Section 4.2.1 and Section 4.2.2). Turning legacy applications into distributed components to make them reusable is an effective way to reduce development costs. CORBA eases the transition from standalone legacy applications to more flexible distributed components to be used over the Internet.

- **Scalability and openness** -- Additional resources can be incorporated into the system as required. This capability should be possible without disrupting the links previously established. In this research, CORBA IDL is used to define all the applications, regardless of the languages, or platforms. CORBA ORB and IIOP make it possible to communicate these applications with the unified interface with each other. New applications with IDL interface could be emerged into the system and work together with previous applications with the same interface. The scalability and openness of the system can easily be extended to various fields.
2. The distributed gear design optimisation system based on the Web-based infrastructure has been developed to provide an integrated gear design environment for distributed integrated gear design across the Internet. Hybrid architecture of the point-to-point structure and the server-centralised model has been developed using CORBA in order to provide a powerful and robust system with the high efficiency and easy-to-manage features. The system has four layers: a user interface layer, application layer, data layer, and the Web server layer. All the gear design programs are encapsulated into CORBA-ware distributed objects, which are located on the application layer. User interface

layer is designed as a unified user interface, through which any remote application objects on the application layer can be invoked. The Web server is used for providing resource administrative services. The data layer contains backend data resources, which are linked to the design applications. The system can be used to integrate distributed computing and design resources across the Internet for distributed computing and collaborative design. The system enables multiple specific domain gear experts and their computerised tools to be integrated to work together. Through the unified graphical user interface, designers can conduct all the integrated design activities including design requirement specification, design optimisation with multiple design interests and multiple design constraint check, viewing the text data and graphical data, and generating the graphical file in exchange format. The system could achieve the strength and fundamental features emphasised in distributed computing: reliability, transparency, interoperability (among heterogeneous environments), scalability, and portability.

3. An optimising tool based on Genetic Algorithms is designed and facilitated in the distributed gear design optimisation system to help designers to leverage multiple design interests. Genetic Algorithms is found to be an appropriate method for conduct complex design optimisation including a large number of variables, multiple design objectives, and numbers of design multiple constraints. It is designed as a CORBA client program through which remote evaluation applications that are designed as CORBA objects can be invoked for the evaluation to each design solution during the evolution procedure. The low level communication between programs provided by CORBA ensures the high efficiency of the system.
4. A 2D gearing gear profiles generation tool is developed and facilitated in the distributed gear design optimisation system. To each solution of design optimisation, a corresponding 2D gearing gear profiles can be produced by the generation tool according to the resultant data. The profiles of the gearing gears can be precisely modelled to further conduct 3D model design and analysis. By using the generation tool a corresponding DXF file can be output. The DXF file, as an exchange graphics file, can be displayed in popular CAD software. In the

integrated design environment, the gears' profiles generation tool helps designer view the design result in time and redesign if it is not ideal.

5. OmniORB is used for implementing the communication between two C++ applications or between a C++ program and a Python program on Linux and Windows. It is found to be the fastest available C++ ORB and completely free. It can be used to implement the communication between C++ on Windows and C++ on Windows, C++ on Linux and C++ on Linux, and C++ on Linux and C++ on Windows.
6. The system provides an integrated environment for resource suppliers to post and share their resources and resource demanders to acquire necessary resources to perform applications in a distributed framework. Design application owners could benefit from the distributed application model. Design program providers who have design and analysis services available could keep their own programs to be only invoked and executed remotely other than downloaded. The application can be used as design services for more users to use. The application codes still reside with owners and the programs execute on previous computers. Only the names of the service objects, the names of the methods on the objects, and input and output information for the methods are published to the clients' developers.
7. The integrated environment is also used for resource demanders to acquire necessary resources to perform extended design applications in a distributed framework. Using the distributed gear design system a designer could conduct gear design through invoke remote design resources to find the rational tuning parameters in consideration of multiple aspects in design and manufacture, within limited time without augmenting his system. The system supports the designer dynamically to choose design objectives that are associated with remote applications. A variable dimensional optimisation allows users to conduct design optimisation of less than or up to 9 variables.
8. The distributed application-developing model CORBA enables it easily to develop the complex distributed design application. The package including integrated gear design optimisation GUI and the optimiser is a sample of complex distributed design application. Heterogeneous design applications that

have been wrapped into CORBA objects can be used as available programming elements within the GUI package. The GUI is deliberate for design visualisation, data analysis, and design collaboration.

9. Optimisation efficiency and optimisation accuracy must be considered. Cascaded Genetic Algorithm mechanism is employed in the GA-based optimising procedure. The solution gained from the first tier is used as the warm value set at the second tier. The smaller space around the warm value set is used as the search space at the second tier. It is found to be a structure to obtain efficiency of coarser search in global space and high accuracy of finer search in smaller space.
10. Bit-field and union structure is used for the encoding of chromosome. It is found to provide a highly efficient data structure for chromosome representation, which simplifies the transformation calculation between binary data and encoding value so that the running efficiency of the program is greatly improved.
11. A new concept of dynamic and variable length chromosome is devised to resolve variable dimensional problems. For those unselected design variables, their corresponding gene segments will not be included in the evolution process. Dynamic mapping array is created to obtain the dynamic combination of genes according to user's variable dimension requirements.
12. Variable penalty function based on simulating annealing algorithm is constructed to deal with the constraints, which provides another approach to penalise the constraint violence. The variable penalty function contains an attenuation factor that is related to the constraints and the temperature schedule in the simulating annealing algorithm. The attenuation factor makes the penalty to the constraint violence from gentle to harsh over the convergence process, so that some infeasible individuals can be kept in the population at the early stage of the evolutionary process to avoid the process to stop prematurely. This work provides an approach to deal with the constraints through tuning the process parameters of convergence instead of tuning the penalty function weighting coefficients.

13. A Web-based system based on the server-centralised structure has been developed, with combination of Servlets, Applets and CORBA. It enables a singular large design program run remotely on the Internet such that the geographically dispersed designers can implement a design on a Web browser and obtain the multiple formats of results from any part of the world. A chain of Servlets are developed, as the extension on Web server side, to retrieve client requirements, activate the large program, monitor the progress of program execution, and view results displayed in multiple formats. A flag data that represents for the status of program execution and Java applet are used to implement the monitoring of the program. CORBA is used to provide the direct communication between C++ legacy algorithm application program and Java Servlet client program. The execution of the program is not affected due to network problem. Multiple user management mechanism enables different users utilise the system simultaneously without the data file conflicts and retrieve the latest design results any time.
14. On-line worm design is developed using Applet/CORBA to provide a plug-in running model of remote program for worm gear design. In this model, a session established between point-to-point can kept linking until stopped manually. Visibroker Gatekeeper is used to establish the communication bridge between CORBA IIOP and HTTP to enable CORBA applet acts as a universal applet to deal with the IIOP messages. This model is used for the flexible need for repetitive parameter modification and interactions with users. Visibroker for C++ and Visibroker for Java are used to develop the communication procedure between Java applet and C++ application on Windows. Java applet could be downloaded on any platform.

9.2 Contributions to Knowledge

The main contribution of this thesis is in constructing a distributed infrastructure to develop Web-based systems for integrated gear design. The resultant Web-based architectures have identified and addressed three main application paradigms.

The development of the infrastructure includes several structural and technical contributions. These contributions can be summarised as follows:

9.2.1 Conceptual Contributions for Web-based Structure

- Heterogeneity of the developed environment enables many different gear design applications, in engineering area, which might be ported in different computers, running on platforms and different operating systems, and written in different languages, can be integrated together and interoperated by each other. It is possible to develop complex distributed application systems for collaborative design over the Internet.
- Reusability of legacy applications provided in the developed system makes those traditional, stand-alone, and single-user computer-aided gear design applications, which are still valuable to perform crucial work, and usually represent a significant investment and years of accumulated experience and knowledge, can be extended for the Web applications using the developed infrastructure without rewriting the essential codes. It is an effective way to reduce distributed application development costs and to ensure the reliable of components.
- A distributed design optimising tool is employed in the Web-based gear design optimisation system where different domain experts and their design applications are distributed. The GA-based tool helps designers to conduct multiple interest evaluations to each solution during the design evolution. This approach of employing a Genetic Algorithms into a Web based environment ensures the effectiveness in collaborative design environment. The low level communication between programs provided by CORBA ensures the high efficiency of the system.
- A Servlet/Applet/CORBA based system in this thesis is developed for a singular large computing program to run remotely on the Internet such that the geographically dispersed designers can implement a design on a Web browser and obtain the multiple formats of results from any part of the world. A flag data that represents for the status of program execution and Java applet are used to implement the monitoring of the program. CORBA

is used to provide the direct communication between C++ legacy algorithm application program and Java Servlet client program. The execution of the program is not affected due to network problem. Multiple user management mechanism enables different users utilise the system simultaneously without the data file conflicts and retrieve the latest design results any time.

- On-line worm design is developed using Applet/CORBA to provide a plug-in running model of remote program for worm gear design. In this model, a session established between point-to-point can kept linking until stopped manually. Visibroker Gatekeeper is used to establish the communication bridge between CORBA IIOP and HTTP to enable CORBA applet acts as a universal applet to deal with the IIOP messages. This model is used for the flexible need for repetitive parameter modification and interactions with users. Visibroker for C++ and Visibroker for Java are used to develop the communication procedure between Java applet and C++ application on Windows. Java applet could be downloaded on any platform.

9.2.2 Technical Contributions

- Cascaded Genetic Algorithm mechanism is employed in the GA-based optimising procedure. It is found to be an effective method to obtain efficiency of coarser search in global space and high accuracy of finer search in smaller space.
- Bit-field and union structure in C++ language is used for the encoding of chromosome. It is found to provide a highly efficient data structure for chromosome representation, which simplifies the transformation calculation between binary data and encoding value so that the running efficiency of the program is greatly improved.
- A new concept of dynamic and variable length chromosome is devised to resolve variable dimensional problems. For those unselected design variables, their corresponding dimension requirements.

- Variable penalty function based on simulating annealing algorithm is constructed to deal with the constraints, which provides another approach to penalise the constraint violence. This work provides an approach to deal with the constraints through tuning the process parameters of convergence instead of tuning the penalty function weighting coefficients.

9.3 Future Work

The work presented in this thesis provides a basis for future research in the Web-based collaborative environment for integrated product design. There are many areas within this research that need further research before the overall goal of developing powerful and robust system for efficiently and effectively supporting total product design via the Internet.

9.3.1 Improving Infrastructure Based on Emerging Web Technologies

In the fast developing IT world, there are some emerging technologies for enlarging the application domain of Web-based architecture. In addition to further implementing the CORBA-based system, exploiting new architecture and new model for design resource integration and communication would be necessary for improving the current collaborative systems.

Web Services

As the extension of this research, a new model utilising the emerging Web Services technology need to be investigated and developed. In Appendix E, a brief overview about Web services, comparison between Web Services and the existing CORBA technology, and an application paradigm of Web Services are presented.

In the future, it is necessary to further exploit the possibilities of CORBA technology and Web Service model applied in design engineering and to develop a hybrid structure for constructing a powerful environment, in order to facilitate the high-efficient features of CORBA and the high-flexible features of Web Service. CORBA's well-coupling model provides a high efficient structure for the need of tightly linked

applications. Web Services technique provides a loosely coupling model for linking those commercial CAD software systems. On the other hand, for the developed CORBA-based system, proper tools can be investigated for the convenient transformation to Web Services.

Grid Technology

Grid computing infrastructures aims at allowing the programmer to aggregate powerful and sophisticated resources scattered around the globe. To achieve this goal, Grid computing community has been making efforts for the creation of advanced services that allow access to high-end remote resources such as batch systems at supercomputing centres, large-scale storage systems, large-scale instruments, and remote applications. The efforts have resulted in the development of Grid services that enable application developers to authenticate, access, discover, manage, and schedule remote Grid resources, but they are often incompatible with commodity technologies [120].

CORBA targets distributed environment and provides transparency on many levels including languages, operating systems, networks, and protocols. It is a possible candidate for applications programmers to develop Grid-based applications. Combination of CORBA structure and Grid domain will allow an easy integration of additional Grid services and functionality within these applications. This model application will be implemented in the future.

XML

The underlying feature behind the flexible-oriented model of Web Services is XML technology, as described in Chapter 9.

As an extension of this work, XML is investigated and studied by the author [121] [122]. CORBA is an enabling technology for creating sophisticated, distributed object systems on heterogeneous platforms. XML is a technology for conveying structured data in a portable way. CORBA allows users to connect disparate systems and form object architectures. XML will allow users to transmit structured information within,

between and out of those systems, and to represent information in a universal way in and across architectures. Both technologies are platform-, vendor- and language-independent. CORBA ties together cooperating computer applications, invoking methods and exchanging transient data that will probably never be directly read by anyone, while XML is intended for the storage and manipulation of text making up humane-readable documents like Web pages. In addition, portable data storage and exchange in XML will relieve CORBA-based systems from low efficiency much data causes in.

In the future work, this model of application will be implemented to provide an environment for integration, communication and dynamic management.

9.3.2 Multiple Disciplinary Optimisations Based on Distributed System

Integrated product development involves collaborative design among multiple disciplinary experts. The interests from these multiple disciplines are comprehensive. The GA-based multiple objective mechanism presented in this thesis provides fundamental structure for solve this problem. All the interests are abstracted into formulised expressions.

However, practical design activities are more complex. Different experts might use different commercial systems (e.g. ANSYS for FEA, Pro-E for Modelling, Matlab for mathematical computing, etc.) and most modelling processes and impacts on design results are complex. An advanced mechanism for supporting design optimisation needs to be further developed, which might contain a smooth link to those commercial CAD systems through parameterisation of complex modelling processes. This hybrid structure enables it possible to construct an intelligent environment.

9.3.3 Semantic CAD Environment

Data exchange technologies such as IGES, DXF and more recently STEP, have significantly helped enable collaboration among disciplines through transferring data across the heterogeneous systems, but this is still a very chunky communication with

low efficiency due to data description at low level of description. Data transferring activities are still ones between low levels of representations and thus causes low efficiency of process flow. Current CAD systems functionally are still at the level of 'drafting' and 'manufacture operation' only.

With the increasing demand in industrial performance improvement, the next generation of collaboration, among CAD/CAM systems, should be highly efficient and support collaborative design. Future research into such a new environment is based on semantic web and high level of representation to the objects, which will be defined in a semantic manner. Intelligent characters received by combining decision-making system will be another differentiating feature to improve the collaborative performance over heterogeneous systems and further improve the design efficiency.

9.3.4 Adaptation of Graphical Data

The system should modify complex data for use in diverse environments including mobile systems with respect to the actual context including device (mobile devices with PDA—Personal Digital Assistant, situation, location etc.). The adaptation is context sensitive and thus each user involved in the collaboration may get data with different representation. It involves the research domains: mobile system structure, variety of expressions and adaptive transformation between them, flexible communication mechanism, and etc.

References

1. K. T. Ulrich and S. D. Eppinger, 2000, *Product Design and Development*, 2nd edit, McGraw-Hill.
2. H. R. Parsaei and W. G. Sullivan, 1993, *Concurrent Engineering*, Chapman & Hall.
3. S. C-Y. Lu, 2004, *Beyond concurrent engineering: a new foundation for collaborative engineering*, CE2004: The 11th ISPE International Conference on Concurrent Engineering Research and Applications, 26-30 July, Beijing, P. R. China.
4. A. Mills, 1998, *Collaborative Engineering and the Internet*, Society of Manufacturing Engineers Dearborn, Michigan.
5. L. Hartley, 1992, *Concurrent Engineering*, Cambridge, MA: Productivity Press.
6. R. M. Hauch, S. W. Jacobs, S. W. Prey, and Heather L. Samsel, 1999, *A distributed software environment for aerospace product development*, AIAA/ASME/ASCE/AHS/ASC Structures Struct Ural Dynamics & Materials Conference, April 12-15, St. Louis Mo USA, 2:1385-1394.
7. S. E. Chrysler, 1992, *Concurrent engineering challenge*, *Manufacturing Engineering*, 108(4): 35-42.
8. L. Wang, 1999, *An approach to collaborative design and intelligent manufacturing*, *Proceedings of International Joint Conference of SCI'99 (Systemics, Cybernetics and Informatics) and ISAS'99 (Information systems Analysis and Synthesis)*, 7:431-437.
9. M. J. Chung, P. Kwon and B. Pentland, 2000, *MIDAS: a framework of integrated design and manufacturing process*, *Proc. Of SPIE Int. Soc. For Optical Eng. Intelligent Systems in Design & Manufacturing*, 4192:348-355.
10. D. Sriram, R. Logcher and S. Fukuda, 1989, *Computer aided cooperative product development* Berlin: Springer.

11. C. Q. Huang and K. L. Mak, 1998, Web-based collaborative product development. *1st International Seminar and Workshop on Engineering Design in Integrated Product Development*, 8-10 October, Wroclaw, Poland.
12. J. W. Erkes, K. B. Kenny and J. W. Lewis, 1996, Implementing shared manufacturing services on the World Wide Web. *Communications of the ACM*, 39(2): 78-87.
13. RaDEO Program Home Page, <http://radeo.nist.gov/radeo/> (accessed on 15/07/2005).
14. AIMS, <http://aims.parl.com/> (accessed on 15/07/2005).
15. TEAM, <http://cewww.eng.ornl.gov/team/home.html> (accessed on 15/07/2005).
16. GEN, <http://urobe.uni-paderborn.de/GEN/> (accessed on 15/07/2005).
17. PRODENT, <http://www.uninova.pt/~prodnet/> (accessed on 15/07/2005).
18. WCIDM, <http://www.admec.ntu.ac.uk/asia-itc/home.html> (accessed on 15/07/2005).
19. C. S. Smith, P. K. Wright, 1996, CyberCut: A World Wide Web based design to fabrication tool, *Journal of Manufacturing Systems*, 15(6): 432-442.
20. M. R. CutKosky, J. M. Tenenbaum and J. Glicksman, MADEFAST: collaborative engineering over the Internet. *Communications of the ACM*, 39(2): 34-45.
21. L. Wang, W. Shen, H. Xie, J. Neelamkavil and A. Pardasani, 2001, Collaborative conceptual design—state of the art and future trends, *Computer-Aided Design* 34: 981-996.
22. S. Rajagopalan, R. Rajamani, R. Krishnaswamy, and S. Vijendran, 2002, *Java Servlet Programming Bible*, Hungry Minds, Inc., New York, USA.
23. P. C. Muller, R. dePooter, J. de Jong, J. M. L. van Engelen. 1996, Using the Internet as a communication infrastructure for lead user involvement in the new product development process. *Proceedings of WET ICE'96*, pages 220-225.
24. A. I. Anton, E. Liang. A Web-based requirement analysis tool. *Proceedings of WET ICE'96*, 1996, pages 238-243.
25. D. S. Kelley, 2001, Web-centric product data management, *Journal of Industrial Technology*, 18(1).

26. G. Toye, M. Cutkosky, J. Tenenbaum and J. Glicksman, 1993, A methodology and environment for collaborative product development, *Proceeding of Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society Press, pages 33-47.
27. M. R. Cutkosky, R. S. Engelmores, R. E. Fikes, M. R. Genesereth, W. S. Mark, J. M. Tenenbaum and J. C. Weber, 1993, PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, 26(1): 28-37.
28. B. Prasad, 1996, *Concurrent Engineering Fundamentals: Integrated Product and Process Organization*, Vol. 1, Prentice Hall, NJ.
29. M. Rezayat, Knowledge-based Product Development using XML and KCs, *Computer-Aided Design* 32(2000a): 199-309.
30. F. Bsharah, M. Less, Requirements and Strategies for the Retention of Automotive Product data, *Computer-aided Design*, 32: 145-158.
31. F. Wang, J. J. Mills, Venkat devarajan, 2002, A conceptual approach managing design resource, *Computers in Industry*, 47: 169-183.
32. Y. X. Xue, 2003, Web-based distributed system and database modelling for concurrent design, *Computer-Aided Design*, 35: 433-452.
33. S. B. Yoo, Y. Kim, 2002, Web based knowledge management for sharing product data in virtual enterprises, *International Journal of Production Economics*, 75(1); 73-183
34. M. J. Chung, P. Kwon and B. Pentland, 2000, MIDAS: A framework of integrated design and manufacturing process, *Proc. Of SPIE Int. Soc. For Optical Eng. Intelligent Systems in Design & Manufacturing*, 4192: 348-355.
35. C. C. Madni and A. M. Madni, 1998, Web-enabled collaborative design process management: application to multichip module design, *IEEE Internatinal conference on Systems Man & Cyberneti Cs*, 11-14 Oct., SAN DIEGO, CA USA, 3(5): 2625-2630.
36. R. Wanger, G. Castanotto, K. Goldberg, *SPIE 1995*, 2596:192-195
37. M. J. Bailey, 1995, Tele-manufacturing: rapid prototyping on the Internet. *TEEE Computer Graphics and Applications 1995*, 20-26 November.
38. K. L. Krause, 1998, Global product data management. In: *Proceedings of 2nd International Conference on Concurrent Engineering: Methods and Tools*.

39. C. Wang, C. Chu and C. Yin, 2001, Implementation of remote robot manufacturing over Internet, *Computers in Industry*, 45(2001):215-219.
40. E. V. Name, G. Egelstein, 2001, Taking a look at Internet-based design in the year 2001. *Electronic Design*, January, pages 42-50.
41. U. Roy, B. Bharadwaj, S. S. Kodkani, M. Cargian, 1997, Product development in a collaborative design. *Concurrent Engng: Res Appl* 1997, 5(4): 347-65.
42. J. Li, H. Zhang, J. Wang, and G. Xiong, 2000, Collaborative design method on network, *JNL. Of Tsinghua University*, 40(9):93-96.
43. H. Wu, H. Zhang, H. Xie and D. Chen, 2000, Web based remote collaborative design system (Cdesign), *Journal of Tsinghua University (Sci & Tech)*, 40(5): 62-65.
44. I-G. Chun and I-S. Hong, 2001, The implementation of knowledge-based recommender system for electronic commerce using Java expert system library, *IEEE Int Symp. On Industrial Electronics Proc.*, 12-16 June, Pusan, pages 1766-1770.
45. A. Ballaminut, C. Colonello, M. Dönszelmann, E. van Herwijnen, D. Köper, J. Korhonen, M. Litmaath, J. Perl, A. Theodorou, D. Whiteson, E. Wolff, 2001, WIRED – World Wide Web interactive remote event display, *Computer Physics Communications*, 140(1): 266-273.
46. R. Raje, M. Boyles, and S. Fang, 1998, CEV: Collaborative Environment for Visualization Using Java RMI, *ACM Workshop on Java for Science and Engineering Computation*.
47. <http://www.informatik.uni-bonn.de/~alda/docs/paperICCCBEX.pdf> 25/01/2005.
48. R. Orfali, and D. Harkey, 1998, *Client/Server Programming with Java and Corba*. Canada: John Wiley & Sons Inc. ISBN 0-471-24578-X.
49. H. Kim, S-B. Yoo, and H-C. Lee, 2000, Web-enabled collaborative design environment, *ETRL JNL*. 22(3): 27-40.
50. J. Li, H. Zhang, J. Wang, and G. Xiong, 2000, Collaborative design method on network, *JNL. Of Tsinghua University*, 40(9): 93-96.
51. U. S. Ong, H. B. Gooi, S. F. Lee, 2001, Java-based applications for accessing power system data via intranet, extranet and internet, *Int. Jnl. Of Electrical Power & Energy Systems*, 23(4): 273-284.

52. C.C. Madni and A. M. Madni, 1998, Web-enabled collaborative design process management: application to multichip module design, *IEEE International conference on Systems Man & Cybernetics*, 11-14 Oct, SAN DIEGO, CA USA, 5(3): 2625-2630.
53. D. W. Denbo and C.R. Windsor, 1999, Oceans MTS/IEEE-Riding the crest, *International Ocean the 21st Century Conference*, 13-16 Sept, SEATTLE USA, 2:1076-2079.
54. K. J. Farrar and J. S. Hwang, CEE Interfacing for Khoros; Visual interactive programming for enterprise research (VIPER), *Proceedings- SPIE The International, Enabling Technology for Simulation Scen CE IV*, 25-27 April, ORLANDO, FL USA, 4026:247-256
55. S. B. Yoo, Y. Kim, 2002, Web based knowledge management for sharing product data in virtual enterprises, *International Journal of Production Economics*, 75(1): 173-183
56. R. M. Hauch, S. W. Jacobs, S. W. Prey, and H. L. Samsel, 1999, A distributed software environment for aerospace product development, *AIAA/ASME/ASCE/AHS/ASC Structures Struct Ural Dynamics & Materials Conference*, 12-15 April, St. Louis Mo USA, 2:1385-1394
57. K-T. Lee, M-L. L. Roh and S. Cho, 2001, Multidisciplinary design optimisation of mechanical systems using collaborative optimisation approach, *International Journal of Vehicle Design*, 25(4):353-368.
58. J. Sang, G. Follen, C. Kim, I. Lopez and S. Townsend, 2001, CORBA wrapping of legacy scientific applications using remote variable scheme, *International Conference of Parallel & Distributed System*, 26-29 Jun, Kyongju
59. <http://www.w3.org/> accessed on 01/Feb/2005
60. Y. Ouyang, M. Tang, J. Lin and J. Dong, 2004, Distributed collaborative CAD system based on Web Service, *Journal of Zhejiang University SCIENCE SCI*, 5(5): 579-586.
61. T. Kostienko, W. Mueller, A. Pawlak, T. Schattkowsky, 2003, An advanced infrastructure for collaborative engineering in electronic design automation, *CE: The Vision for the Future Generation in Research and Applications*, R. Jardim-Goncalves et al. (eds) Sets & Zeitlinger, Lisse, ISBN 90 5809 622 X.

62. K. Hodota, 1997, Product and information life-cycle with product configuration database system on Internet, *in 21 st Century Commerce & CALS EXPO International Conference '97*.
63. A. McKay, F. Erens, and M. S. Bloor, , 1996, Relating product definition and product variety, *Research in Engineering Design*, 2:63-80.
64. O. A. Suarez, J. L. A. Foronda, and F. M. Abreu, 1998, Standard based framework for the development of manufacturing control system, *International Journal of Computer Integrated Manufacturing System*, 11:401-415.
65. J. K. Wu, T. H. Liu, and G. W. Fischer, 1992, An integrated PDES/STEP based information model for CAE and CAM applications, *The Second International Conference on Automation Technology*, pages179-187.
66. Y. M. Chen and Y. T. Hsiao, 1997, A collaborative data management framework for concurrent product and process development, *International Journal of Computer Integrated Manufacturing System*, 10:446-449.
67. R. Jordim-Goncalves et al., Implementaiton of computer integrated manufacturing systems using SIP: CIM case studies using a STEP approach, *International Journal of Computer Integrated Manufacturing systems using SIP: CIM case studies using a STEP approach, International Journal of Computer Integrated Manufacturing System*, 10.
68. Y. V. R. Reddy et al., 1993, Computer support for concurrent engineering, *Computer*, 26:12-16.
69. G. L. Smith and J. C. Muller, 1994, PreAmp—a pre-competitive project in intelligent manufacturing technology: an architecture to demonstrate concurrent engineering and information sharing, *Concurrent Engineering: Research and Applications*, 2:107-115.
70. M. S. Bloor, J. Owen, 1991, CAD/CAM product-data exchange: the next step, *Computer-aided Design*, 23:237-243.
71. NIST, Industrial Automation Systems – <http://www.nist.gov/> (accessed on 15/07/2005).
72. M. P. Bhandarkar, B. Downie, M. Harwick, R. Nagi, 2000, Migrating from IGES to STEP: one to one translation of IGES drawing to STEP drafting data, *Computers in Industry*, 41:261-277.

73. P-Y. Chao and Y-C. Wang, 2001, A data exchange framework for networked CAD/CAM, *Computer In Industry*, 44:131-140.
74. Y. Oh, S-H. Han, H. Suh, 2001, Mapping product structures between CAD and PDM systems using UML, *Computer-Aided Design*, 33(2001):521-529.
75. Y. P. Zhang, C. C. Zhang, H. P. B. Wang, 2000, An Internet based STEP data exchange framework for virtual enterprises, *Computers in Industry*, 41(2000):51-63.
76. M. Hardwick, D. L. Spooner, T. Rando, K. C. Morris, 1996, Sharing manufacturing information in virtual enterprises, *Communications of ACM* 39 (1996) 46-54.
77. W. C. Regli, 1997, Internet-enabled computer-aided design. *IEEE Internet Computing*, January.
78. E. Ly, 1997, Distributed JAVA applet for project management on the Web, *IEEE Internet Computing*, 2(1997):21-26.
79. D. Brown, K. Wersprille, 1997, The OCAI initiative (open Cax architecture and interoperability), *Proceedings of CALS Expo. International 1997*, Tokyo, pages 37-41.
80. T. Kimuro H. Akasaka, Y. Nishino, New approach for searching distributed databases by agent technology, *Proceedings of the CALS Expo. International 1997*, Tokyo, pages 13-22.
81. P-Y. Chao, Y-C. Wang, 2001, A data exchange framework for networked CAD/CAM, *Computers in Industry*, 44(2001):131-140.
82. K-S. Chin, Y. Zhao and C. K. Mok, 2002, STEP-based multiview integrated product modelling for concurrent engineering, *The International Journal of Advanced Manufacturing Technology*, 20:896-906.
83. Y. X. Xue, Web-based distributed system and database modelling for concurrent design, *Computer-Aided Design*, 35:433-452.
84. H. V. Leong, K. S. Ho, and W. Lam, 2003, Web-based Workflow Framework with CORBA, *Concurrent Engineering: Research and Applications, Vol 2*.
85. S. C. F. Chan, T. Dillon and V. T. Y. Ng, 2003, Exchanging STEP Data Through XML-based Mediators, *CE CONCURRENT ENGINEERING: Research and Applications*, March, 11(1).

86. H. Kim, J-Y. Lee, S-B. Han, 1999, Process-centric distributed collaborative design, *Proceedings of 1999 ASME DETC, DETC99/EIM-9081*, Las Vegas, Nevada.
87. H. Kin, S-B. Yoo, and H-C. Lee, 2000, Web-enabled collaborative design environment, *ETRI Journal*, 27-40 September, 22(3).
88. M. P. Case, S. C. Y. Lu, 1996, The discourse model for collaborative engineering design, *Computer-Aided Design* 28(5):333-345.
89. X. Li, X. Zhou and X. Ruan, 2001, Research on key techniques of collaborative design system, *High Technology Letters*, 7(2):51-53.
90. A. M. Malkawi, R. S. Scrivivasan, Y. K. Yi and R. Choudhary, 2003, Performance-based design evolution: the use of genetic algorithms and CFD, *Eight International INPSA Conference*, August 11-14, Eindhoven, Netherlands.
91. F. Hoffmann and G. Pfister, Automatic Design of Hierarchical Fuzzy Controllers Using Genetic Algorithms, <http://www.nada.kth.se/~hoffmann/eufit94.pdf>, 02/Feb/2005.
92. AGMA 2101-C95, 1995, Fundamental rating factors and calculation methods for involute spur and helical gear (Metric version), American Gear Manufacturers Association.
93. DIN 868, publication:1976-12 General definitions and specification factors for gears, gear pairs and gear trains.
94. ISO 6336-1: 1996 Calculation of load capacity of spur and helical gears—Part 1: Basic principles, introduction and general influence factors.
95. ISO 6336-2: 1996 Calculation of load capacity of spur and helical gears—Part 2: Calculation of surface durability (pitting).
96. ISO 6336-3: 1996 Calculation of load capacity of spur and helical gears—Part 3: Calculation of tooth bending strength.
97. ISO 6336-5: 1996 Calculation of load capacity of spur and helical gears—Part 5: Strength and quality of materials.
98. BS 436: Part 3, 1986, Method of Calculation of Contact and Root Bending Stress Limitations for Metallic Involute Gears, British Standards Institution.
99. http://www.engineersedge.com/gear_design.htm, 05/Jun/2005

100. I. C. Parmee, 2001, *Evolutionary and Adaptive Computing in Engineering Design*, Springer-Verlag London Limited.
101. *MAAG Gear Company LTD*. MAAG Gear Book CH-8023 Zurich, Switzerland.
102. D. W. Dudley, 1992, *Dudley's Gear Handbook McGraw-hill Book Company*, Inc., New York.
103. M. Gen and R. Cheng, 1997, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, Inc. Canada.
104. J. H. Holland, 1992, *Adaptation in Nature and Artificial Systems*. MIT Press.
105. A. Kurpati, S. Azarm and J. Wu, 2002, Constraint handling improvements for multiobjective genetic algorithms, *Structural and Multidisciplinary Optimisation*, 23:204-213
106. <http://www.w3.org/TR/2002/WD-ws-arch-20021114/#roles> (accessed on 15/07/2005).
107. http://en.wikipedia.org/wiki/Inter-process_communication (accessed on 15/07/2005).
108. <http://www.oasis-open.org/home/index.php> (accessed on 15/07/2005).
109. <http://www.w3.org/> (accessed on 15/07/2005).
110. <http://www.ws-i.org/> (accessed on 15/07/2005).
111. http://en.wikipedia.org/wiki/Simple_Object_Access_Protocol (accessed on 15/07/2005).
112. <http://java.sun.com/xml/jaxrpc/index.jsp> (accessed on 15/07/2005).
113. <http://www.xmlrpc.com/> (accessed on 15/07/2005).
114. <http://www.webopedia.com/TERM/X/XML.html> (accessed on 15/07/2005).
115. <http://www2002.org/CDROM/alternate/395/> (accessed on 15/07/2005).
116. <http://www.capeclear.com/> (accessed on 15/07/2005).
117. <http://www.idc.co.za/> (accessed on 15/07/2005).
118. G. Laszewski, M. Parashar, S. Verma, J. Gawor, K. Keahey and N. Rehn, A CORBA commodity Grid kit, <http://www.caip.rutgers.edu/TASSL/Papers/corbacog-ccpe-gce01.pdf> (accessed on 15/07/2005).
119. S. Ji and D. Su, 2005, A heterogeneous collaborative design environment with dynamic management features, *Proceeding of the 9th International Conference*

- on Computer Supported Cooperative Work in Design*, Coventry, 24-26 May, United Kingdom
120. S. Ji, D. Su and J. Li, 2005, Integration, management and communication of heterogeneous resources based on Web technologies, *Lecture Notes in computer Science* (accepted).
 121. Andrews, G.G. and Argent, J. D. (1992). Computer-Aided Optimal Gear Design, International Power Transmission and Gearing Conference, vol. 1, ASME.
 122. EL-Bahloul, A.M.M. (1992). CAD for Gears: Spur, Helical and Double Helical Gears, *Mansoura University Journal* 17(3).
 123. Jau regui, J.C. and Sanalvador, R. L. (1992). Software for Optimum Gear Design, *Mechanical Design and Synthesis, DE-Vol. 46*, ASME.
 124. Metwalli, S.M. and El-danaf, E.A. (August 1996). CAD and Optimisation of Spur and Helical Gear Set. Proceedings of ASME Design Engineering Technical Conferences and Computer in Engineering Conference, Irvine, California.
 125. Yiu-Wing, C. and Siang-Kok, S. (1998). An Expert System for Gearing Design Application, *International Journal of Computer Applications in Technology*, Vol 11-26.
 126. EI-Sayed Aziz and C. Chassapis, Knowledge-based Geometry Generation for Spur and Helical Gears, *Concurrent Engineering: Research and Applications*, Vol 10 (3): 251-261.
 127. K. Mehlhorn, 1999, LEDA-a platform of combinatorial and geometric computing, Cambridge University Press.
 128. A. Riedl, 2001, A CORBA/XML-based architecture for distributed network planning tools, ICEIS 2001, Juli, Setubal.

Appendix A The Main Features of CORBA

The main features of CORBA are:

- ORB Core
- OMG Interface Definition Language (OMG IDL)
- Interface Repository
- Language Mapping
- Stubs and Skeletons
- Dynamic Invocation and Dispatch
- Object Adapters
- Inter-ORB Protocols

Most of these are illustrated in Figure 3.4, which shows the components of CORBA relate to one another. Each component is described in detail below.

A.1 The Object Request Broker (ORB)

ORB is the central component in CORBA. It acts as a mediator between client requests and object implementations. ORB provides mechanisms for invoking methods on local or remote objects. These mechanisms automate the search for object location, object creation, activation and object management. Besides that, ORB also provides means for message exchange between objects. ORB is defined as a logical set of processes. Any ORB implementation that provides the appropriate interface is acceptable. Different ORBs may have different implementations extending from client-resident ORBs to server-based ORBs, ORBs that are part of an operating system or simple library-based ORB.

As can be seen in Figure 3.3, an ORB has two different interfaces; it communicates with the client through the client interface and with the server through the server interface. If a client wants to make a request, it can do so through the ORB's client interface. Similarly, when the ORB passes this request to the server (object implementation), it uses its server interface.

The key feature of the ORB is the transparency of how it facilitates client/object communication. Ordinarily, the ORB hides the following: object location, object implementation, object execution state, and object communication mechanisms.

A.2 The Interface Definition Language (IDL)

The development of flexible distributed applications on heterogeneous platforms requires a strict separation of interfaces from implementations. Such separation ensures:

- Platform independence and
- Language independence

An Interface Definition Language (IDL) helps to accomplish this separation. OMG IDL is an object-oriented language used for defining interfaces to objects in CORBA. These interfaces are then used by clients to access object implementations.

An interface definition written in OMG IDL completely defines the interface of an object and fully specifies the parameters of each object's method (operation). An OMG IDL interface provides the information needed to develop clients that use the object's operations. Clients (and servers) are not written in OMG IDL, which is purely a descriptive language, but in languages for which so-called "mappings" from OMG IDL have been defined. There are many mappings already available, such as the mappings to C, C++ and SmallTalk, Java, Ada95, COBOL, Modula 3.

A.3 Static Stubs and Skeletons

In addition to generating programming language types, OMG IDL language compilers and translators also generate client-side stubs and server-side skeletons. A stub is a

mechanism that effectively creates and issues requests on behalf of a client, while a skeleton is a mechanism that delivers requests to the CORBA object implementation.

After an IDL file is written, it is passed to an IDL compiler that creates source code for both the client and the server.

The OMG has specified official language mappings for IDL to Java, C, C++, Smalltalk, and COBOL. Thus, the IDL compiler supplied by the vendor does the work of appropriately mapping the IDL definitions to their appropriate languages types.

A.4 Dynamic Invocation and Dispatch

In addition to static invocation via stubs and skeletons, CORBA supports two interfaces for dynamic invocation:

- *Dynamic Invocation Interface* (DII) – which supports dynamic client request invocation;
- *Dynamic Skeleton Interface* (DSI) – which provides dynamic dispatch to objects.

Static stubs and skeletons must be generated at build time and compiled in with your source code. With the DII and the DSI, it is not necessary to use IDL to generate static stubs and skeletons. Rather, the DII provides clients an interface by which they can dynamically query the ORB's Interface Repository for available objects and construct method requests on-the-fly. The Interface Repository is a standard CORBA component, a container of CORBA interfaces that are implemented by servers in the current environment. Similarly, the server object does not need to be compiled in with a static skeleton to receive requests. The DSI automatically enables new objects to receive request without having inherited from the IDL generated skeleton.

A.5 Object Adapters

The final subcomponent of CORBA, the Object Adapter (OA), serves as the glue between CORBA object implementations and the ORB itself. It is the main way by

which Object Implementations access services via the ORB. The OA serves the following several important functions for the object implementation:

- Generation and mapping of object references to their implementations
- Registrations of implementations
- Activation and deactivation of object implementation

A.6 Inter-ORB Protocols

The General Inter-ORB Protocol (GIOP) and the IIOP are protocols specified in CORBA to provide the interoperability between the different ORBs provided by different ORB vendors. GIOP specifies the format of the messages and a common representation of the data that will be transferred among the ORBs. This protocol was designed to run over a connection-oriented transport layer protocol, as in the case of TCP/IP. IIOP defined how GIOP messages are to be transmitted over the TCP/IP protocol. In addition, IIOP also provides the means to use the Internet as a powerful communication mechanism between ORBs since this protocol allows interaction among remotely distributed ORBs.

Appendix B Java Platform Independence

Java is platform independent at both the source and the binary level. Platform independence at the source level means that you can move Java sources files from system to system and compiled and executed cleanly on any systems. Java compiled binary files are also platform independent and can run on multiple platforms (if they have a Java virtual machine available) without the need to recompile the source.

Normally, when a program written in C++ or in most other languages is compiled, the compiler translates the program into machine code or processor instructions. Those instructions are specific to the processor the computer is running – so, for example, if a program code is compiled on an Intel-based system, the resulting program will run only on other Intel-based systems. If the same program is needed to use on another system, the developer has to go back to the original source code, get a compiler for that system, and recompile the code so that the program is made specific to that system. Figure B.1 shows the result of this system: multiple executable for multiple systems.

In contrast, things are different when source code is written in Java. The Java development environment actually has two parts: a Java compiler and a Java interpreter. The Java compiler takes the Java program and, instead of generating machine codes from the source files, it generates bytecodes. Bytecodes are instructions that look a lot like machine code, but are not specific to any one processor.

To execute a Java program, it is needed to run a program called a bytecode interpreter, which in turn reads the bytecodes and executes the Java program (see Figure B.2). The Java bytecode interpreter is often also called the Java virtual machine or the Java runtime.

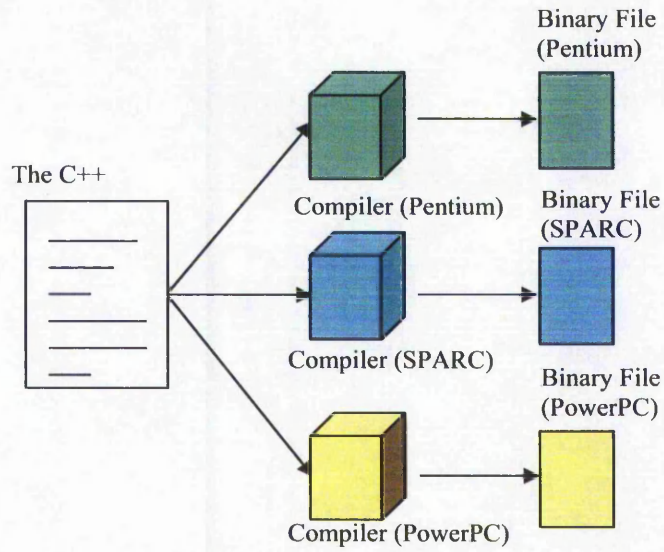


Figure B.1 Traditional compiled programs

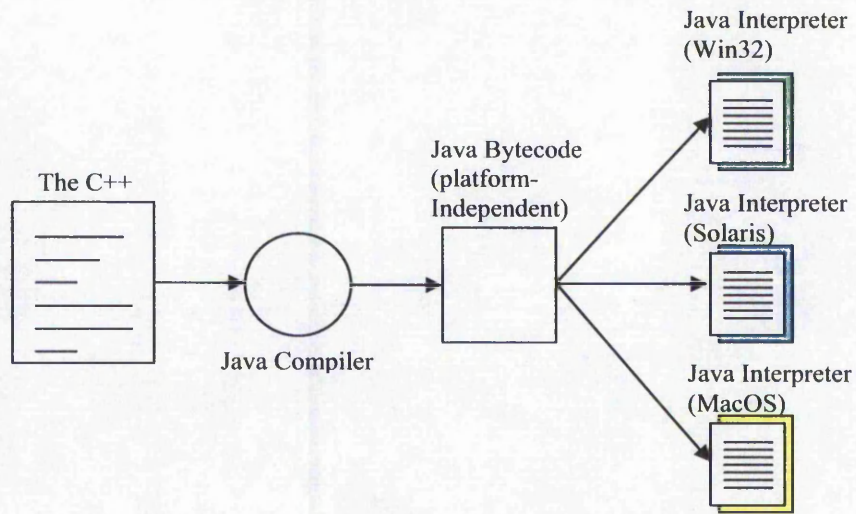


Figure B.2 Java programs

In this application, Java applications and Java applets are used for the development of client-side user interfaces in the client/server system, and Java Servlets are used for the development of the service in the server-side.

Appendix C Involute Spur and Helical Gear Design

A gear is a toothed wheel that is usually, but not necessarily, round. The teeth may have any of an almost infinite variety of profiles. The purpose of gearing is to transmit motion from one shaft to another. This motion transfer may be accompanied by changes in direction, speed, and shaft torque.

Gear design is one of the classical topics of mechanical engineering design. The classical route followed for the design of gears is to appeal to standards, like BS, AGMA, DIN or ISO [92~98]. These standards are based on extremely large collections of results and empirical rules from practical experience in a vast range of engineering applications. They provide a set of formulae, rules and charts to design the gearing taking into account various working conditions and several aspects of their performance, such as the power level, noise, lubrication conditions, wear rate, likelihood of impact, pitting, and corrosion. Nevertheless, actual gear design involves very difficult and complex calculations and trial and error, and thus often requires an iterative process to determine those design parameters that would satisfy performance and strength requirements, which would result in an efficient and reliable operation for gear transmission system.

In this project, gear design problem is used as case study to explore the possibility of proposed collaborative system for improving design efficiency. In this chapter, overview of basic knowledge of gears is first given, and then relative knowledge, formulae, and experience to gear performance, gear strength, and specific sliding are presented. Especially gear geometrical design with rack shift and its constraint

conditions are identified. The calculations use procedures, algorithms and data from standards ANSI, ISO, DIN, BS and specialised literature.

C.1 Basic Knowledge of Gears

C.1.1 Gear Type

In general, gears may be divided into three classifications based on the arrangement of the axes of the gear pair, e.g. parallel (spur, helical, and Internal spur or helical), perpendicular (crossed helical, cylinder worm gearing) or intersect (bevel), for the different requirement of application. Table C.1 gives features and their application environment of the common gears [99].

Table C.1 Gears applications

Type	Precision rating	Features	Applications	Comments regarding precision
Spur	Excellent	Parallel shafting, high speeds and loads, highest efficiency	Broadly applicable to a variety of applications and wide range of velocity ratios	Simplest tooth elements offering maximum precision. First choice, recommended for all gear meshes, except where very high speeds and loads or special features of other types, such as right – angle drive, cannot be avoided.
Helical	Good	Parallel shafting. High speeds, high loads	Most applicable to high speeds and loads. Also used wherever spurs and are used.	Equivalent quality to spurs except for complication of helix angle. Recommended for all high – speed and high – load meshes. Axial thrust component must be accommodated.
Internal (spur & helical)	Fair	Parallel shafts High speeds High loads	Internal drives requiring high speeds and high loads; offers how sliding and high stress loading; good for high capacity, long life. Used in planetary gears to produce large reduction ratios.	Not recommended for precision meshes because of design of design, fabrication, and inspection limitations. Should only be used when internal features is necessary.

Bevel	Fair to good	Intersecting shafts High speeds High loads	Suitable for 1:1 and higher velocity ratios and for right – angle meshes (and other angles)	Good choice for right – angle drive, particularly low ratios. However, complicated tooth form and fabrication limits achievement of precision.
Crossed helical	Poor	Skewed shafting Point contact High sliding Low speeds Light loads	Relatively how velocity ratio; low speeds and light loads only. Any angle skew shafts.	To be avoided for precision meshes. Point contact limits capacity and precision. Suitable for right – angle drives if light load. A less expensive substitute for bevel gears. Good lubrication essential because of point contact and high sliding action.
Cylindrical Worm Gearing	Fair to good	Right – angle skew shafts; high velocity ratio; high speeds and loads; low efficiency; most designs; nonreversible	High velocity ratio Angular meshes High loads	Worm can be made to high precision, but worm gear has inherent limitations. To be considered for average precision meshes, but can be of high precision with care. Best choice for combination high velocity ratio and right – angle drive. High sliding requires excellent lubrication.

In this project, involute spur and helical gears, as shown in Figure C.1, are used as the case of collaborative design.

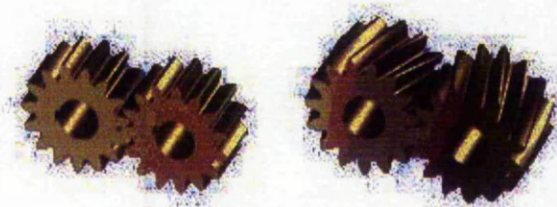


Figure C.1 Spur and helical gears

C.1.2 Properties of Involute

An involute is the locus of a point on a straight line which rolls without slipping round the circumference of a stationary circular disc, see Figure C.2. The stationary circular

disc represents the base circle for the involute tooth profile. The parametric equations for the polar coordinates of the involute are:

$$r_k = r_b / \cos \alpha_k \tag{C-1}$$

$$\theta_k = \text{inv} \alpha_k = \tan \alpha_k - \alpha_k \tag{C-2}$$

α_k is the pressure angle at a point K on the involute, angle between the force F and velocity v_k . r_b is the radii of the base circle.

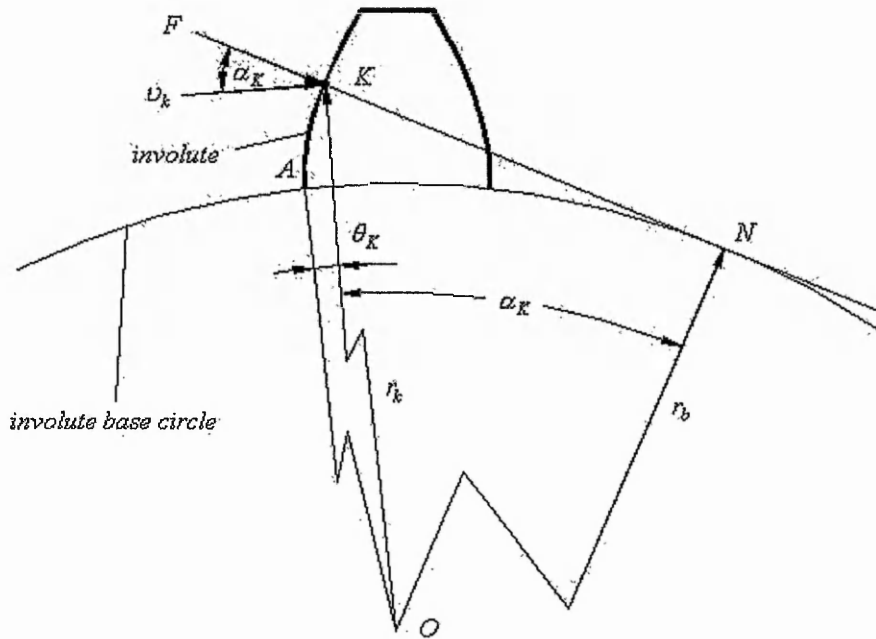


Figure C.2 Involute curve of gear profile

C.1.3 Basic Geometrical Parameters of Gears

The design of the geometry substantially affects a number of other parameters such as functionality, safety, durability or price. In this section, basic parameters of gears are identified.

The involute is used for most spur and helical gears. As shown in Figure C.3, the basic geometrical parameters of a spur gear with involute profile are described below.

Number of teeth z

Determination of the optimal number of teeth is not an unambiguous task and cannot be solved directly. Numbers of teeth affect mesh conditions, noise, efficiency and production costs. Therefore, the number of teeth is chosen and specified according to qualitative and strength indices.

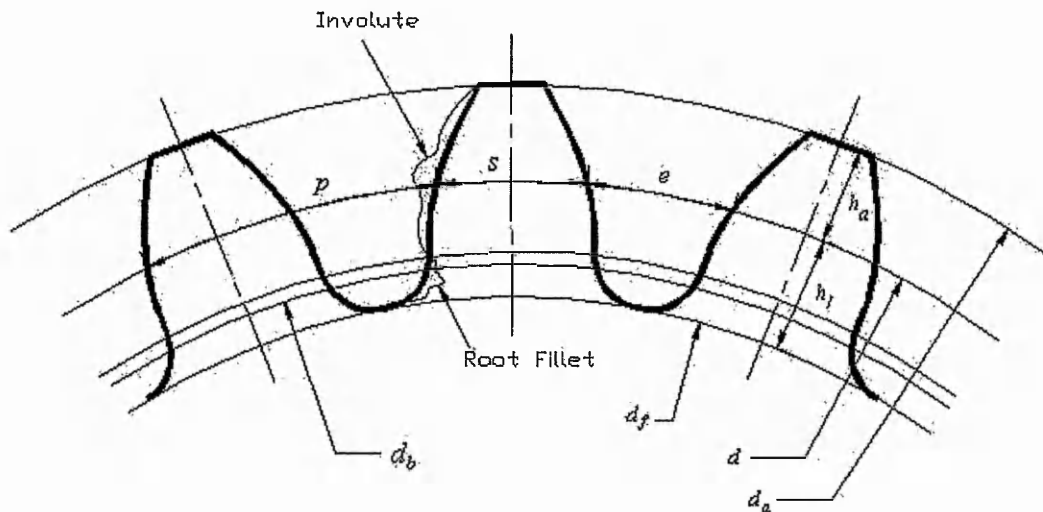


Figure C.3 Spur gear geometry

A generally applicable rule states that increasing the number of teeth (with the same axis distance) leads to:

- increase in loading capacity of the surface (contact, seizure, wearing)
- improvement in the gearing coefficient
- decrease in loading capacity in bend
- reduction in production costs

The rule is that higher numbers of teeth are chosen for higher output powers and lower transmission ratios.

Normal pressure angle α

Pressure angle represents the angle at which the force is transmitted between the teeth. The normal pressure angle is measured from the tangent of the reference circle. This angle also determines parameters of the basic profile and is standardised to an angle of

20° . Other normal pressure angles are also employed for special applications, e.g. 15° , 17.5° , 22.5° and 25° .

$\alpha = 15^\circ$ is for certain printing machinery and kinematically exacting gear drives, such as for the movement of telescopes or radar reflectors.

$\alpha = 17.5^\circ$ is for marine gears with deep teeth where particularly quiet running is required.

$\alpha = 22.5^\circ$ and 25° is for cases where the flanks are subjected to extremely high contact stresses.

Changing the angle has a variety of effects upon the tooth and the stresses acting upon it. Increasing the pressure angle makes the tooth thicker and increases the radii of curvature at the pitch line. The effects of this will be to improve the bending strength, increasing loading capacity in contact and wear. An additional effect of the increasing the pressure angle is that it reduces the contact ratio.

Module (m or m_n):

The module represents the ratio of the reference circle diameter to the number of teeth on the gear and therefore, defines the size of the teeth. The lower the module, the smaller the teeth and thus the higher the stresses acting on them for the same power transfer. It is generally applicable that for a higher number of teeth it is possible to use a smaller module and vice versa.

The module has been standardised. The standard modules could be selected from BS 436-2:1970, as shown in Table C.2. The standard modules to the helical gears are the normal modules. Series 1 represents preferred modules, and series 2 the second choice modules.

Table C.2 Standard normal modules

Series 1	1 1.25 1.5 2 2.5 3 4 5 6 8 10 12 16 20 25 32 40 50
Series 2	1.125 1.375 1.75 2.25 2.75 3.5 4.5 5.5 7 9 11 14 18 22 28 36 45

The module is one of basic parameters of gears, which is used to calculate other geometrical size. In the following formulas, module is used to calculate other geometrical parameters.

Diameter of reference circle

$$d = m * z \tag{C-3}$$

Pitch on reference circle

$$p = \pi m \tag{C-4}$$

Base diameter

$$d_b = mz \cos \alpha \tag{C-5}$$

Addendum of the generated gear

$$h_a = h_a^* m \tag{C-6}$$

Dedendum of the generated gear

$$h_f = (h_a^* + c^*)m \tag{C-7}$$

where h_a^* - *Addendum coefficient of the generated gear*, $h_a^* = h_{aP}^*$

h_{aP}^* - *Rack addendum coefficient*

Helix angle β

The helix angle is the inclination of tooth to the direction rotation. The range of the angle is practical up to a limit of 45°. Increasing the angle has the effect of increasing the contact ratio. However increasing the helix angle also has the effect of increasing the axial load possibly to a point where the deflection of the shaft is intolerable or the size of bearings will be too great.

Spur gear ($\beta = 0$) is used with slow speed and highly loaded gearing. Helical gear ($\beta > 0$) is used with high speed gearing; it is characterised by lower noise and higher loading capacity, enabling the use of a lower number of teeth without undercutting.

For the helical gear, as shown in the Figure C.4, all the calculations of the tooth profile on the transverse section are the same as on spur gear, but the normal parameters such as m_n , h_{an}^* , or α_n are standardised. The relationship between transverse and normal parameters is given by the below.

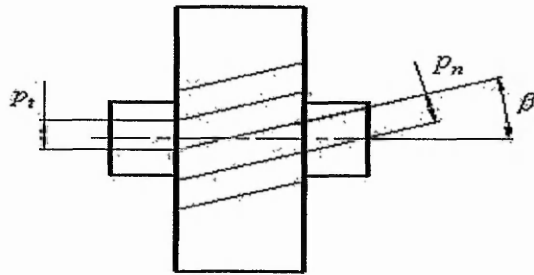


Figure C.4 Geometrical parameters of helical gear

$$m = m_n = m_t \cos \beta \tag{C-8}$$

$$h_a^* = h_{an}^* = h_{at}^* / \cos \beta \tag{C-9}$$

$$c^* = c_n^* = c_t^* / \cos \beta \tag{C-10}$$

$$\tan \alpha = \tan \alpha_n = \tan \alpha_t \cos \beta \tag{C-11}$$

Face width ratio

$$\phi_d = \frac{b}{d_1} = \frac{b \cos \beta}{m z_1} \tag{C-12}$$

Face width ratio is the ratio of the width of the gear to the diameter ($\phi_d = b/d$) and determines the load distribution across the gear face. The upper limit of this ratio can be determined according to the Table C.3, relating to material properties, heat treatment and application.

Table C.3 Face width ratio

Gear Surface	Gear mounting relative to bearings	
	Symmetric	Asymmetric

<i>Hardness < 180 HB</i>	< 1.0	< 1.0
<i>Hardness > 180 HB</i>	< 1.0	< 1.0
<i>Induction or Case Hardened</i>	< 1.0	< 0.9
<i>Nitrided</i>	< 0.8	< 0.6

C.1.4 Basic Rack Profile

The basic rack profile is fundamental to the specification of involute gears. It determines the tooth profile on the gear, the generating rack profile and associated rack shaped tools. The relationship between them is shown in Figure C.5.

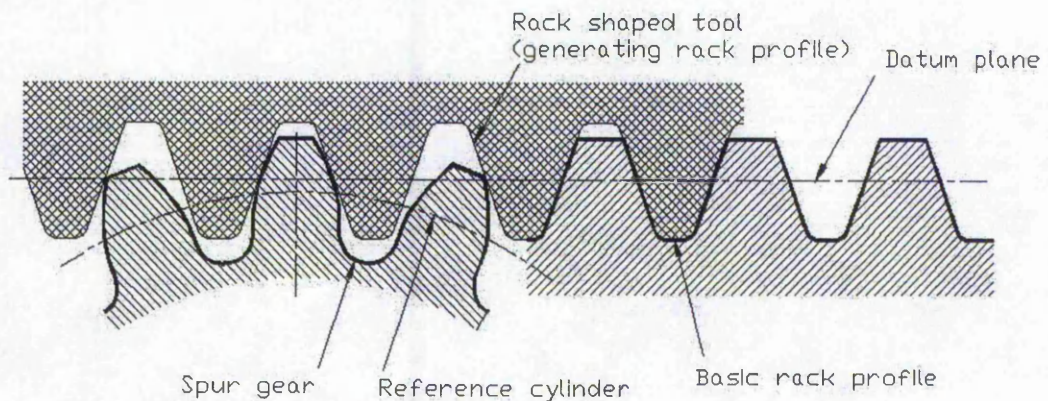


Figure C.5 Relation between spur gear, basic rack and rack shaped tool

With the standard rack profile, the tooth thickness is equal to the tooth space width at the profile datum line and hence equal to half the pitch, gear and mating gear can be cut with the same tool. The entire dimensions for defining the basic rack profile must be contained in the tooth data. The parameters relevant to rack are listed as below.

Rack addendum h_{aP}

Rack dedendum $h_{fP} \quad h_{fP} = (h_{aP}^* + c_P^*)m$

The fillet radius coefficient ρ_{fP}^*

The fillet radius $\rho_{fP} \quad \rho_{fP} = \rho_{fP}^* m$

The rack tip radius is related to the root profile of the generated tooth and is used during calculation of the tooth's performance. The root profile is positioned within the clearance region of the dedendum between the root circle and the involute of the tooth, as shown in Figure C.3.

The bottom clearance c_p determines the greatest possible fillet radius $\rho_{fp_{\max}}m$. The fillet radius $\rho_{fp_{\max}}m$ however must not be greater than that resulting in a full fillet root. The condition for this is:

$$\rho_{fp_{\max}} = \frac{c_p}{1 - \sin \alpha} \leq \left(\frac{\pi m}{4} - h_{fp} \tan \alpha \right) \tan \left(\frac{90^\circ + \alpha}{2} \right) \quad (\text{C-13})$$

C.2 Addendum Modification

C.2.1 Addendum Modification

When gears are produced by a generating process, the datum line of the basic rack need not necessarily form a tangent to the reference circle. The tooth form can be shifting the datum line from the tangential position. The involute shape of the tooth profile is retained, and the effect is merely to use parts further from or nearer to the origin of the same involute. The radial displacement from the tangential position is termed addendum modification. The displacement is considered positive when in the direction away from the centre of the gear and negative when nearer towards the centre.

C.2.2 Application Types of Addendum Modification

a) Standard gearing ($x_\Sigma = x_1 = x_2 = 0$)

This can be thought of as a specific case of addendum modification, as shown in Figure C.6a, with the gearing angle α' equalling to the pressure angle on the reference circle α and the centre distance a' equalling to standard centre distance a .

This type of transmission has the features of simple design, convenient usage and easy –to-wear due to the weak gear root.

b) Height profile shift ($x_\Sigma = x_1 + x_2 = 0, x_1 = -x_2$)

As shown in Figure C.6b, with this type of transmission the centre-distance remains the same as for the standard gears and is not affected by the profile shift and gearing angle α' is equal to the pressure angle on the reference circle α . Only addendum and dedendum height is changed.

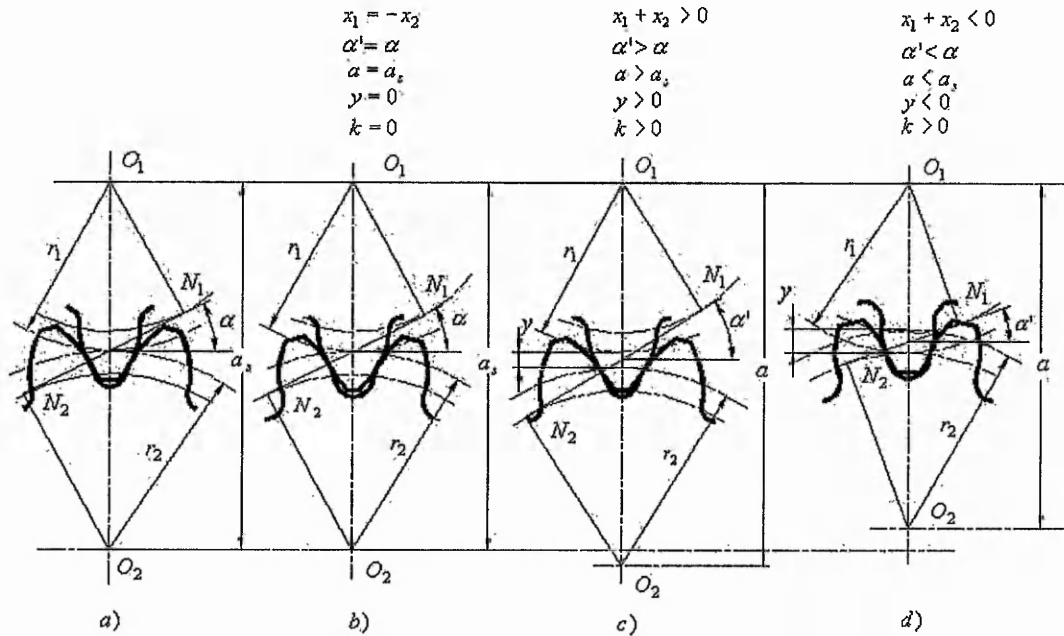


Figure C.6 Application types of addendum modification

$$h_{a1} = (h_a^* + x_1)m \quad h_{f1} = (h_a^* + c^* - x_1)m \quad (\text{C-14})$$

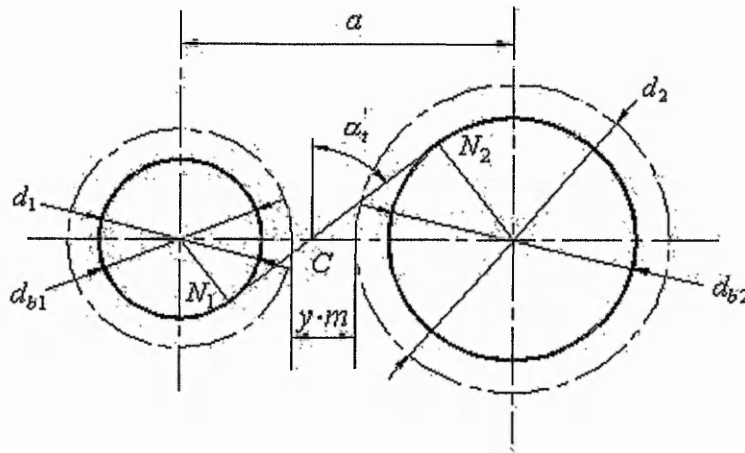
$$h_{a2} = (h_a^* - x_1)m \quad h_{f1} = (h_a^* + c^* + x_1)m \quad (\text{C-15})$$

As $x_\Sigma = 0$, The positive shift the one gear (usually the pinion, as it has the lower number of teeth and will therefore, reduce the possibility for undercutting) is equal to the negative shift of the mating gear, $x_\Sigma = x_1 + x_2 = 0$.

This is generally suitable for the need of smaller structural size. Meanwhile the load carrying capacity of the pinion is improved and the balance distribution of pinion and wheel strength can be achieved.

c) Angle profile shift -- positive gearing ($x_{\Sigma} = x_1 + x_2 > 0$)

As shown in Figure C.6c, the working angle α'_i is greater than the pressure angle on the reference angle of standard gears, e.g. the generating pressure angle α_i , and the centre distance is greater than the standard centre distance $a' > a$. The amount by which the centre distance deviates from the sum of the reference circle radii is known as the centre distance modification $y \cdot m$ and the working pressure angle α'_i is given by the formula below and is shown in Figure C.7. y is called as the centre distance modification coefficient.


 Figure C.7 Mating gears with centre distance modification $y \cdot m$

$$y \cdot m = a' - a = \frac{d_1 + d_2}{2} \left(\frac{\cos \alpha_i}{\cos \alpha'_i} - 1 \right) \quad (\text{C-16})$$

$$\text{where } \tan \alpha_i = \frac{\tan \alpha}{\cos \beta}, \quad \alpha_i = \arctan \left(\frac{\tan \alpha}{\cos \beta} \right)$$

$$\text{inv} \alpha'_i = \text{inv} \alpha_i + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha, \quad \alpha'_i = \text{arcinv} \left(\text{inv} \alpha_i + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha \right)$$

$u = z_2 / z_1$, representing the transmission ratio.

For mating external spur and helical gears, the centre distance modification is always smaller than the sum of the addendum modifications. An addendum shortening of $k \cdot m$ is necessary to maintain is therefore necessary to maintain the basic rack profile bottom clearance c_p . k is called as the addendum shortening coefficient. In the case of positive transmission, because of $x_\Sigma > 0$, then $y > 0$, $k > 0$.

$$k = x_\Sigma - y = \frac{(z_1 + z_2)}{2} * \frac{\text{inv}\alpha_t' - \text{inv}\alpha_t}{\tan \alpha} - \frac{(z_1 + z_2)}{2 \cos \beta} * \left(\frac{\cos \alpha_t}{\cos \alpha_t'} - 1 \right) \quad (\text{C-17})$$

This type of modification allows for a fixed centre distance to be achieved when different to the standard and achieve the smaller structural size, the slighter wearing of gear teeth, and the improved load carrying capacity.

d) Angle profile shift -- negative ($x_\Sigma = x_1 + x_2 < 0$)

As shown in Figure C.6d, with this type of modification both the reference circles are crossed, $\alpha_t' < \alpha_t$, $a' < a$, $y < 0$, but $k > 0$. This is used for the fixed centre distance with certain degree of decrease on the load carrying capacity.

C.2.3 Necessary Calculations for Addendum Modification

C.2.3.1 Addendum coefficient h_a^*

Addendum coefficient, h_a^* , defines the length of the tooth in terms of the module, as illustrated in Figure C.3. The addendum (h_a) is taken as the region from the reference circle to the tip of the tooth and the dedendum (h_f) as the region from the reference circle to the root of the tooth.

For the gears with profile shift the addendum and dedendum are given by the formula below.

$$k = x_\Sigma - y = \frac{(z_1 + z_2)}{2} * \frac{\text{inv}\alpha_t' - \text{inv}\alpha_t}{\tan \alpha} - \frac{(z_1 + z_2)}{2 \cos \beta} * \left(\frac{\cos \alpha_t}{\cos \alpha_t'} - 1 \right) \quad (\text{C-18})$$

$$\text{where } \tan \alpha_t = \frac{\tan \alpha}{\cos \beta}, \alpha_t = \arctan \left(\frac{\tan \alpha}{\cos \beta} \right)$$

$$\text{inv} \alpha'_t = \text{inv} \alpha_t + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha, \alpha'_t = \text{arcinv} \left(\text{inv} \alpha_t + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha \right)$$

$u = z_2 / z_1$, representing the transmission ratio.

$$h_{a1} = m_t (h_{at}^* + x_{t1}) - k * m_t \quad (\text{C-19})$$

$$h_{f1} = m_t (h_{at}^* + c^* - x_{t1}) \quad (\text{C-20})$$

$$h_{a2} = m_t (h_{at}^* + x_{t2}) - k * m_t \quad (\text{C-21})$$

$$h_{f2} = m_t (h_{at}^* + c_i^* - x_{t2}) \quad (\text{C-22})$$

where $m_t = m_n / \cos \beta = m / \cos \beta$

$$h_{at}^* = h_{an}^* \cos \beta = h_a^* \cos \beta$$

$$c_i^* = c_n^* \cos \beta = c_n^* \cos \beta$$

$$x_{t1} = x_{n1} \cos \beta = x_1 \cos \beta$$

$$x_{t2} = x_{n2} \cos \beta = x_2 \cos \beta$$

Generally, $c^* = 0.25$, $h_a^* = h_{ap}^* = 1$ is standard and used for certain applications. In

this project, $h_a^* = h_{ap}^* = (0.75, 1.25)$ are another two options.

$h_a^* = h_{ap}^* = 0.75$ is for stub teeth for gear tooth couplings.

$h_a^* = h_{ap}^* = 1.25$ is for marine gears with deep teeth.

C.2.3.2 Centre Distance

$$a = \frac{d_{b1} + d_{b2}}{2 * \cos \alpha'_t} = \frac{(d_1 + d_2) \cos \alpha_t}{2 * \cos \alpha'_t} = \frac{1}{2} \frac{m z_1}{\cos \beta} (1 + u) \frac{\cos \alpha_t}{\cos \alpha'_t} \quad (\text{C-23})$$

$$\text{where } \tan \alpha_t = \frac{\tan \alpha}{\cos \beta}, \alpha_t = \arctan \left(\frac{\tan \alpha}{\cos \beta} \right)$$

$$\text{inv} \alpha'_t = \text{inv} \alpha_t + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha, \alpha'_t = \text{arcinv} \left(\text{inv} \alpha_t + \frac{2(x_1 + x_2)}{z_1(1+u)} \tan \alpha \right)$$

$u = z_2 / z_1$, representing the transmission ratio.

If the centre distance of a gear is given, then the following expression is considered as an equality constraint. The variable x_2 is not independent variable but determined by the variable x_1 and given a .

C.2.3.3 Assessment of Bending Stress

Bending stress calculations

$$f_A = \sigma_{FP} = \frac{2\sigma_{F0}(\sigma_B - \sigma_K)Y_N Y_R Y_X Y_M Y_\delta}{(\sigma_B + \sigma_{F0} Y_N Y_R Y_X) S_{F \min}} s \quad (C-24)$$

Actual calculated bending stress

$$\sigma_F = \frac{F_t}{bm_n} * Y_F Y_S Y_\beta K_A K_V K_{F\alpha} K_{F\beta} \quad (C-25)$$

The permissible torque

$$T_{FP} = \frac{60 * 10^3}{2\pi} * \frac{P_{FP}}{n_1} \quad (C-26)$$

Peak torque capacity for bending stress

$$\frac{T_{F \max}}{T_{FP}} = K_A K_V * \frac{\sigma_{FY} Y_M Y_S Y_{\delta stat}}{\sigma_{FP} S_{F \min}} \quad (C-27)$$

Permissible core bending stress

$$\sigma_{FPcore} = \frac{(\sigma_{Bcore} - \sigma_{Rcore}) Y_M Y_N Y_X}{(1 + 0.5 Y_N Y_X) S_{F \min}} \quad (C-28)$$

Actual core bending stress

$$\sigma_{Fcore} = \sigma_F * \left(1 - \frac{2c_{eff}}{M_n}\right) * \frac{Y_{Sred}}{Y_S} \quad (C-29)$$

Permissible power capacity and torque based on bending stress

$$P_{FP} = \frac{bd_1 n_1 m_n}{60 * 10^6 / \pi} * \frac{1}{Y_F} * \frac{1}{Y_S} * \frac{1}{Y_\beta} * \frac{1}{K_A} * \frac{1}{K_V} * \frac{1}{K_{F\alpha} K_{F\beta}} * \sigma_{FP} \quad (C-30)$$

Nominal tangential force for bending stress

$$F_{Ft} = \frac{2000T_{F1}}{d_1} \quad (C-31)$$

$$F_{Ft} = \frac{1000P_H}{v} \quad (C-32)$$

where, m_n , normal module; F_t , nominal tangential force at reference circle; $Y_{F,S,B,FY,M,S,\delta stat,M,N,X,Sred}$, factors related to tooth properties and working condition; $Y_{F,S,B,FY,M,S,\delta stat,M,N,X,Sred}$, load factors; σ_{F0} , bending endurance limit; σ_B , ultimate tensile strength; σ_R , residual stress; S_F , minimum demanded safety factor.

C.2.3.4 Contact stress

Permissible contact stress

$$\sigma_{HP} = \sigma_{Hlim} Z_L Z_V Z_R Z_M Z_W Z_X Z_N / S_{Hmin} \quad (C-33)$$

Actual contact stress

$$\sigma_H = Z_H Z_E Z_\epsilon \sqrt{\frac{F_{Ht}}{bd_1} * \frac{(u+1)}{u} * K_A K_V K_{H\alpha} K_{H\beta}} \quad (C-34)$$

Permissible power capacity and torque based on contact on contact stress P_{HP} and T_{HP}

$$P_{HP} = \frac{bd_1^2 \pi n_1}{60 * 10^6} * \frac{u}{(u+1)} * \frac{1}{Z_H^2 Z_\epsilon^2} * \frac{1}{K_A} * \frac{1}{K_V} * \frac{1}{K_{H\alpha} K_{H\beta}} * \left(\frac{\sigma_{HP}}{Z_E} \right)^2 \quad (C-35)$$

$$T_{HP} = \frac{60 * 10^3}{2\pi} * \frac{P_{HP}}{n_1} \quad (C-36)$$

Peak torque capacity for contact stress

$$\frac{T_{Hmax}}{T_{HP}} = K_A K_V \left(\frac{\sigma_{HY} Z_M}{\sigma_{HP} S_{Hmin}} \right)^2 \quad (C-37)$$

$$F_{Ht} = \frac{2000T_{H1}}{d_1} \quad (C-38)$$

$$F_{Ht} = \frac{1000P_H}{v} \quad (C-39)$$

C.2.3.5 Specific Coefficient at Both the Pinion and Wheel Gear

Gear teeth both roll and slide on one another. Rolling velocity is beneficial because it entrains lubricant between contact areas, increases oil film thickness, and reduces

severity of asperity contacts. Sliding velocity on the other hand generates heat from friction and increases asperity distress.

The specific sliding ratio is the ratio between the sliding velocity and the absolute velocity in the tangent plane at the contact point:

$$\eta_1 = (v_{t2} - v_{t1}) / v_{t1} \tag{C-40}$$

$$\eta_2 = (v_{t1} - v_{t2}) / v_{t2} \tag{C-41}$$

The specific sliding ratio ranges from zero at the rolling point to a negative value in the addendum and to a positive value in the dedendum, as shown in Figure C.8.

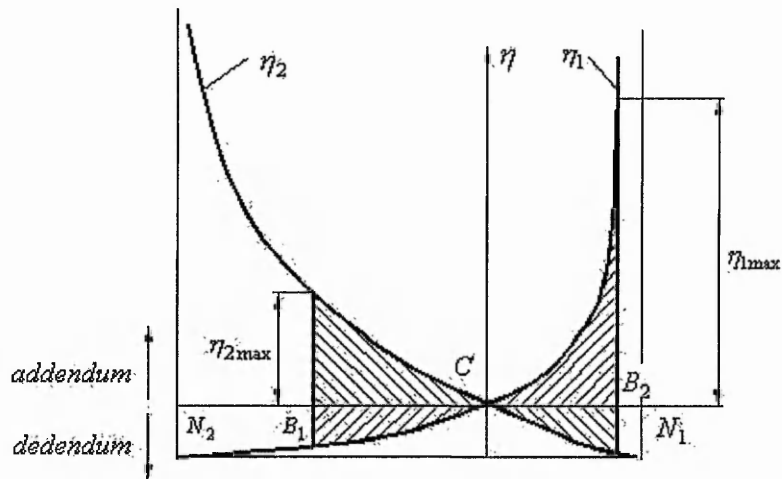


Figure C.8 Slide specific sliding ratio

$$\eta_{1\max} = \frac{\tan \alpha_{a2} - \tan \alpha'}{(1 + z_1 / z_2) \tan \alpha' - \tan \alpha_{a2}} \left(\frac{u + 1}{u} \right) \tag{C-42}$$

$$\eta_{2\max} = \frac{\tan \alpha_{a1} - \tan \alpha'}{(1 + z_2 / z_1) \tan \alpha' - \tan \alpha_{a1}} (u + 1) \tag{C-43}$$

Experiments show areas with positive specific sliding ratio are significantly more prone to micropitting than areas with negative ratio, and a low specific sliding ratio is beneficial to micropitting resistance.

From the figure if actual action line B_1B_2 is moved left through addendum shift modification, $\eta_{1\max}$ could be reduced, through choosing a proper addendum modification coefficient, $\eta_{1\max}$ and $\eta_{2\max}$ will be equal.

C.2.3.6 Constraint Conditions

Determination of the geometry of the gear and tooling teeth for functional design objectives must be constrained under some conditions otherwise will cause the failure of design. These constraint conditions includes the check for the mating interference, the undercutting check of tooling rack, calculation of tooth thickness at the tip, bending strength and contact strength analysis, and so on.

1. Constraints on strength

$$\text{Contact strength } \sigma_{HP} \geq \sigma_H, \quad (\text{C-43})$$

$$\text{Bending strength, } \sigma_{FP} \geq \sigma_F \quad (\text{C-44})$$

Permissible contact strength σ_{HP} should be much than actual contact strength σ_H , permissible bending strength σ_{FP} should be much that actual bending strength σ_F .

2. Cutter interference condition

Cutter interference, as shown in Figure C.9, results when during its exit from the tooth space the end point of the straight of the straight flank of the generating rack profile generates a trochoidal fillet, which intersects the involute. Part of the involute is cut away thereby with consequent reduction in the usable profile.

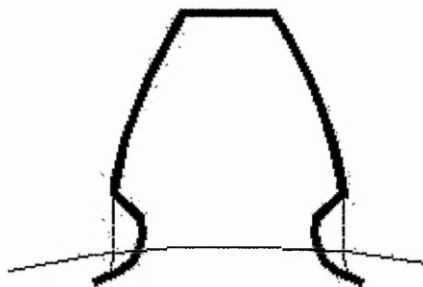


Figure C.9 Undercutting occurring

The minimum addendum modification coefficient x which avoids cutter interference on spur gears is given by:

$$x_{1,2} \geq h_{ap} + \frac{1}{2 \cos \beta} z_{1,2} \sin^2 \alpha_t \quad (\text{C-45})$$

3. Tooth tip thickness

$$S_{a1,2} \geq 0.3m \quad (\text{C-46})$$

4. Interference on mating external spur and helical gears

Interference is caused by contact with non-involute parts of the tooth flank of the mating gear with consequent severe shocks in the transmission. The most frequent cause is faulty design geometry of the mating gears. If the design follows the guide lines for external spur and helical gears given below, a subsequent check for interference is not necessary.

Interference at the roots of mating external gear teeth

$$g_{inv} - g'_{inv} \leq 0, \quad (\text{C-47})$$

$$g_{inv} = \frac{d_{b1}}{2} * \tan \alpha_t - \frac{h_{inv}}{\sin \alpha_t}, \text{ where } h_{inv} = m \left[h_{JP}^* - \rho_{JP}^* (1 - \sin \alpha) - x_1 \right]$$

$$g'_{inv} = a * \sin \alpha'_t - g_{a2}, \text{ where } g_{a2} = \frac{d_{b2}}{2} * \tan \alpha_{a2} \quad \cos \alpha_{a2} = \frac{d_{b2}}{d_{a2}}$$

Interference at the tips of mating external spur and helical gear teeth

$$g_{a2} - a * \sin \alpha'_t \leq 0, \quad (\text{C-48})$$

$$K_A - 1 \leq 0, \quad (\text{C-49})$$

$$K_E - 1 \leq 0 \quad (\text{C-50})$$

5. Slide/roll ratio for the gear tooth tips

$$\text{Slide/roll ratio at pinion tooth tip } \frac{K_A}{1 - K_A} \leq 3 \quad (\text{C-51})$$

$$\text{Slide/roll ratio at gear tooth tip } \frac{K_E}{1 - K_E} \leq 3 \quad (\text{C-52})$$

$$\text{where } K_E = \frac{u+1}{u} \left(1 - \frac{\tan \alpha'_t}{\tan \alpha_{a1}} \right), \text{ con} \alpha_{a1} = \frac{d_{b1}}{d_{a1}}$$

$$K_A = \frac{u+1}{u} \left(1 - \frac{\tan \alpha'_t}{\tan \alpha_{a2}} \right), \text{ con} \alpha_{a2} = \frac{d_{b2}}{d_{a2}}$$

Rack tip fillet radius coefficient limit checking

$$\rho_{fp} \leq \frac{0.25}{1 - \sin \alpha} \quad (\text{C-53})$$

6. Variables x_1 , x_2 and fixed centre distance a

If a fixed centre distance is given, x_1 and x_2 are not independent any more. During the evolution procedure, x_2 is determined by the evolution parameter x_1 and the fixed centre distance. The calculation is given by the below.

$$x_1 + x_2 = \frac{z_1 + z_2}{2} * \frac{\text{inv} \alpha'_t - \text{inv} \alpha_t}{\tan \alpha}$$

$$x_2 = (x_1 + x_2) - x_1$$

7. Rack tip radius coefficient ρ_{fp} and α

Because $\rho_{fp} \cdot m \leq \frac{c_p \cdot m}{1 - \sin \alpha}$, where c_p is taken as 0.25 (typical standard value), ρ_{fp} is dependent on the pressure angle α .

- a) If α is to be optimised, then ρ_{fp} cannot be given a fixed value and must be optimised.
- b) If the value of α is given, then ρ_{fp} must be given in a proper value, determined by the above equation. In the program developed, the associate relationship of α and ρ_{fp} has been predefined.

8. Total contact ratio ε

This is the sum of transverse contact ratio ε_α and overlap ratio ε_β

$$\varepsilon = \varepsilon_\alpha + \varepsilon_\beta = \frac{1}{2\pi} [(z_1(\tan \alpha_{at1} - \tan \alpha'_t) + z_2(\tan \alpha_{at2} - \tan \alpha'_t))] + B \sin \beta / \pi m_n \quad (\text{C-54})$$

For smooth meshing of gears, it is necessary that the other pair of teeth enters in meshing before the first pair is released. The contact ratio in the face plane says how many teeth are in meshing simultaneously. With the value $\varepsilon = 1$ this corresponds to a limit case when only one pair of teeth is in meshing at the given moment. With the value $\varepsilon = 2$, there are two teeth in meshing simultaneously. In case the value is between $1 < \varepsilon < 2$, the meshing will include partly one pair of teeth and partly two pairs. The transverse overlap ratio is applicable in the case of helical gearing (angle $\beta > 0$). Contact ratio ε must always be higher than 1.2.

$$\varepsilon \geq 1.2 \quad (\text{C-55})$$

C.3 Summary

Geometry design of gear with addendum modification (profile addendum) involves strength analysis, cutting check when manufacturing, motion interfere check, contact ratio check, specific sliding calculation, and so on. Multiple design criteria involved causes in the design difficulty.

Appendix D Basic Methods Controlling the GA Process

D.1 Chromosome Representation

In order to tackle a problem using a GA, candidate solutions must be encoded in a suitable form. The most used way of encoding is a binary string. A chromosome then could look like the below:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution.

Of course, there are many other ways of encoding. The encoding depends mainly on the solved problem. For example, one can encode directly integer or real numbers, sometimes it is useful to encode some permutations and so on.

D.2 Selection, Crossover and Mutation

Selection operator

Parents are selected according to their fitness. Therefore the calculation of the fitness is a critical stage within the GA and has the greatest effect upon guiding the search. The fitness is the measure of how successful the information encoded within the genome has been toward achieving an optimum solution. The calculation that determines the fitness of the parameters within the genome is termed the fitness function and consists of a single or combination of calculations, depending upon the optimisation requirements. It is these requirements that govern the type and nature of the fitness

function, which is unique to the application the GA is being applied to. The aims and requirements of the search must be clearly identified and the functions modelled to simulate these goals.

Chromosomes are selected from the population to be parents for crossover. The problem is how to select these chromosomes. According to Darwin's theory of evolution the best ones survive to create new offspring. The probabilistic method determines the probability of reproduction for each chromosome based upon its fitness in relation to the rest of the population. The better the chromosomes are, the more chances to be selected they have. The selection process is based upon the associated probability of the genomes and a random factor. There are many methods in selecting the best chromosomes, which are roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. Examples in this study are roulette wheel selection.

Imagine a **roulette wheel** where all the chromosomes in the population are placed. The size of the section in the roulette wheel is proportional to the value of the fitness function of every chromosome - the bigger the value is, the larger the section is, as shown in Figure D.1.

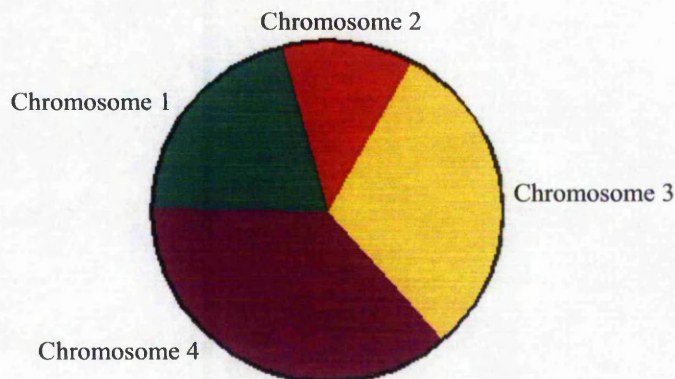


Figure D.1 Parent selection roulette wheel

Crossover operator

Crossover operates on selected genes from parent chromosomes and creates new offspring. The simplest way how to do that is to choose randomly some crossover

point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent.

Crossover can be illustrated as follows: (| is the crossover point):

Chromosome 1	11011 00100110110
Chromosome 2	11011 11000011110
Offspring 1	11011 11000011110
Offspring 2	11011 00100110110

The number of chromosomes participating depends upon some preset crossover probability which establishes the percentage of the population that will be selected as parents.

There are other ways to make crossover, for example in this project more crossover points are chosen. Crossover can be quite complicated and depends mainly on the encoding of chromosomes. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

Multi-point crossover uses two or more points to divide up the genome. The number of the points can be fixed, while the location of these points can be set randomly, as shown below

Chromosome 1	011 001 01011101 010010
Chromosome 2	011 011 01010010 101001
Offspring 1	011 011 01011101 101001
Offspring 2	011 001 01010010 010010

The crossover rate (denoted by p_c) is defined as the ratio of the number of offspring produced in each generation to the population to the size (usually denoted by pop_size). This ratio controls the expected number $p_c \times pop_size$ of chromosomes to undergo the crossover operation. A higher crossover rate allows exploration of more of the solution space and reduces the chances of settling for a false optimum; but if this

rate is too high, it results in the wastage of a lot of computation time in exploring unpromising regions of the solution space.

Mutation operator

The action of mutation is relatively simple. In a binary representation mutation merely flips the randomly selected binary digit from zero to one or vice versa, as shown below

01100101011101010010
01100101010101010010

In genetic algorithms, the mutation operator introduces new genetic material into the population thereby avoiding stagnation of the genetic pool whilst constantly sampling widely varying areas of the search space. Its purpose is to maintain diversity within the population and inhibit premature convergence.

A random mutation probability is pre-set at the beginning of the GA run. The mutation rate (denoted by p_m) is defined as the percentage of the total number of genes in the population. The mutation rate controls the rate at which new genes that would have been useful are never tried out; but if it is too high, there will be much random perturbation, the offspring will start losing their resemblance to the parents, and the algorithm will lose the ability to learn from the history of the search.

The level of crossover and mutation is decided on the basis of experience with GAs and trial and error. However the region is generally from 70% to 95%. Multiple random crossover points have been used in this investigation during reproduction of the chromosomes, due to the large numbers of chromosomes and their non-uniform size. The purpose of the mutation is to increase the search area and prevent local optimisation. The rate of mutation is varied, ranging from 0.0001~0.1 in an attempt to increase the repeatability of the results. Determining the level of mutation is achieved by trial and error.

D.3 Population

In addition to the crossover rate and the mutation rate, another factor of the GA that has a dramatic effect upon the optimisation route and success is the size of the population. The size of the population represents the number of search cases that are being performed with combinations of parameter values. Therefore, the larger the population is, the more comprehensive the cover of the search area. However, there must be a limit to the maximum and minimum sizes of the population. Calculable methods for determining the population size for any application have not been found and are therefore, set through trial and error and with consideration of two factors; computational speed and minimum cover. The maximum size of population has no limit, except for the limitations of the computational speed, the larger the population the more calculations to perform and the longer the convergence, due to the wider the population size. The minimum cover takes into consideration the number of optimisation parameters and their ranges. The outcome of which, will be a population size that will offer enough variation within the initial genes to adequately cover the search area.

D.4 Termination of the Optimisation Process

An important part of the genetic algorithm is the termination criteria i.e. when to stop reproducing the individuals. Frequency of testing for termination criteria is situational and set according to individual preference. There are two methods of terminating the GA that have been identified and investigated for their application to this project, fixed length, a specified fitness of the population, and individual identity.

The process towards convergence is driven by fitness and there is a convenient termination criterion. In fact, after many generations of evolution via the repeated application of reproduction, crossover, and mutation, the individuals in the population will often begin to look alike. At this point, the GA typically terminates because additional evolution will produce little improvement in fitness. Many termination criteria may be used, in which the most simple is to just stop after some predetermined number of generations, i.e. the fixed length method.

This is a crude method of control as the process may either continue unnecessarily after the optimum has been achieved or more importantly terminate prematurely. Using this method requires extensive trial and error testing and analysis of the results to determine a suitable length increasing development time considerably. However, this method can be used to limit the optimisation process when convergence upon a single solution is not necessary.

Another termination criteria can be a specified average fitness of the population. When the specified fitness becomes the average fitness of the population the genetic algorithm can be terminated.

Convergence criteria form an adaptive approach to the termination of the optimisation process. This method compares the parameters of each genome with others from within the population. The comparison will identify if the population's genomes are identical. If this is the case, convergence has been achieved and the process terminates. The principle behind the method is based upon the survival of the fittest process that the GA applies. As the generations increase and the optimum solution develops the stronger genes will pass through crossover more frequently due to the weak genes being removed from the population. Therefore, the number of identical genomes within the population will increase. Once the entire population is identical the optimum achievable solution and convergence will have been obtained. It is not practical to continue the generations until the population total converges. For this reason and to decrease the time taken, a convergence limit is set. The convergence limit corresponds to the percentage of the population that must be identical before termination of the optimisation process. Setting the limit to allow for the maximum percentage of mutations prevents the possibility of an infinite loop developing. However, the higher the convergence level setting, the longer the process will take. Therefore, it is necessary to trade off the possibility of achieving a global maximum against process time. This compromise is achieved through experience and trial and error and has been determined from their use in this research as between 50% and 80%.

The process of comparing the genomes can be performed in two ways, one relating to the structure of the genome (with respect to the bit formation), the other relating to the decoded information that is contained within each gene. The first method checks each bit within the genome, basing the convergence check on the complete genome composition being identical. This method comprehensively covers the search space irrespective of if limits have been imposed on the parameters during decoding. The second method checks the decoded parameters extracted from the genome enabling limits that are imposed.

Appendix E CORBA Model Transition towards Web Services

E.1 Introduction

Distributed object computing has become increasingly popular as more complex products are written using a multi-tier architecture. A number of products and protocols are available for facilitating communication, and many developers have trouble deciding which ones to use in a given situation. Many of communication methods work well together, and each has its strengths and weaknesses.

As the extension of this research, a new emerging developing model, Web Services technology, has been investigated and studied. In this chapter, a brief overview about Web services is first given, followed by the comparison between Web Services and the existing CORBA technology, and an application paradigm of Web Services is finally presented.

E.2 Web Service

Web Service is an emerging distributed middleware technology, regarded as a competing middleware technology of CORBA, which uses simple XML-based protocols and standards to allow applications or systems to exchange data across the Web [108]. Software applications written in various programming languages and running on various platforms can use Web services to exchange data over computer networks like the Internet in a manner similar to Inter-process communication (IPC) on a single computer [109]. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards. OASIS (Organization for the Advancement of Structured Information Standards) [110] and the W3C (The World Wide Web Consortium) [111] are the steering committees

responsible for the architecture and standardization of Web services. To improve interoperability between Web service implementations, the WS-I (Web Services Interoperability Organisation) [112] has been developing a series of profiles to further define the standards involved.

All data to be exchanged is formatted with XML tags. Services are described in terms of the messages accepted and generated. The encoded message is transmitted through a transport protocol such as SOAP (Simple Object Access Protocol) [113], JAX-RPC (Java API for XML-based RPC, RPC means Remote Procedure Calling) [114] or XML-RPC [115]. Users of such services do not need to know anything about the details of the implementation (object model, programming language, etc.); they only need to be able send and receive messages.

Web services aims to give a new computing paradigm based on a loosely coupled service-oriented architecture. Direct machine to machine interaction that was hitherto deemed infeasible is now possible due to the rapid technological advances in XML and SOAP technologies. There have been, however, much confusion about the Web services paradigm and many arguments about if it is a real alternative of CORBA.

Web Services technologies are based on open standards recommended by the World Wide Web Consortium (W3C). A Web service is a software system identified by a Uniform Resource Identifier (URI), whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols [108].

The core components of Web Services standards are WSDL (Web Services Description Language), SOAP (the Simple Object Access Protocol), and UDDI (Universal Description, Discovery and Integration), which are all based on XML (eXtensible Markup Language). The component architecture for Web services is illustrated in Figure E.1.

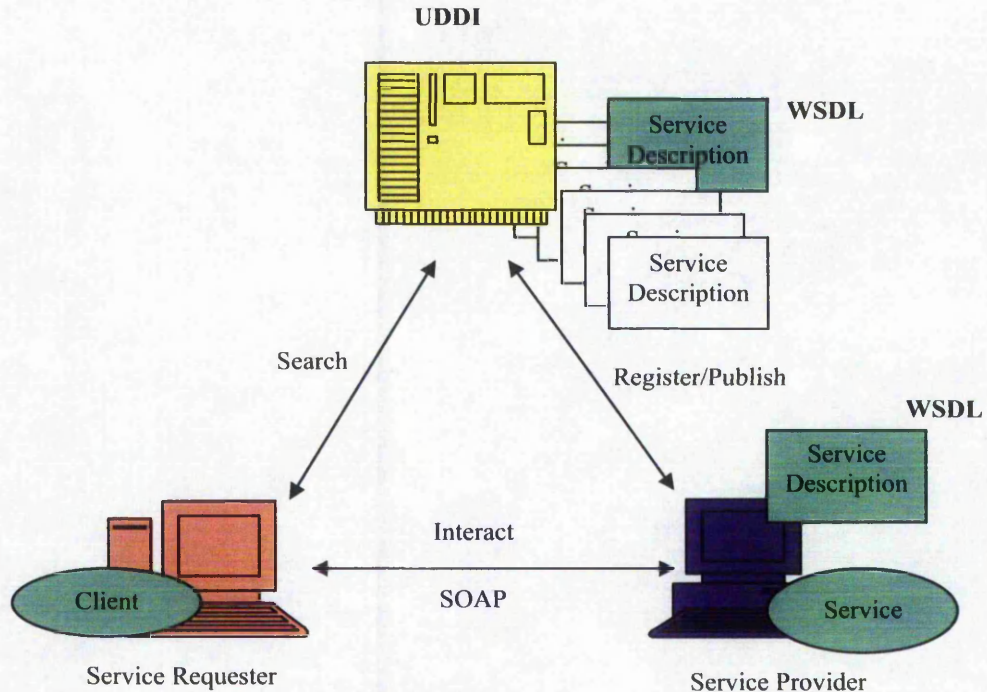


Figure E.1 Component architecture in a Web Service

The specifications (i.e., interface) of services can be described using WSDL, a metadata language that defines how service providers and service requesters communicate with Web Service applications. It is an XML schema that describes where a service is located, what operations are supported, and the format of the messages to be exchanged based on how the service is invoked, regardless of what message formats or network protocols are used for communication.

SOAP is a communication protocol for exchanging information in a decentralised, distributed environment. It defines a mechanism to pass commands and parameters between clients and services. Like Web Service as a whole, SOAP is independent of the platform, object model, and programming language being used. In addition to SOAP, there are other transport protocols such as JAX-RPC and XML-RPC.

A UDDI server provides a “meeting place” for Web Services, an information database of Web Services, which stores descriptions about service owners and the services they offer in a common XML format. Just as businesses list their products and services in a

telephone directory, UDDI is used for providers to register and publish their services that requesters can then discover and invoke. Web-based applications interact with UDDI registries, which are kept in sync.

The Extensible Markup Language (XML) is a pared-down version of SGML (Standard generalised Markup Language), designed especially for Web documents. It allows designers to create their own customised tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations [116]. It is the key to all the other Web Service standards and promises to simplify and lower the cost of data interchange and Web publishing.

E.3 Comparisons between Web Services and CORBA

In both CORBA and Web services, the interactions between a client process and an object server are implemented as object-oriented RPC-style (Remote Procedure Calling) communications. For a typical RPC structure, to invoke a remote function, the client makes a call to the client stub. The stub packs the call parameters into a request message, and invokes a transport protocol to ship the message to the server. At the server side, the transport protocol delivers the message to the server stub, which then unpacks the request message and calls the actual function on the object. In DCOM, CORBA, and Web services, the client stub and the server stub are referred as different name. Table E.1 illustrates these different names for the concepts.

Table E.1 Client and server components in different RPC architecture

RPC Architectures	CORBA	DCOM	Web Services
Client Stub	Stub	Proxy	Service Proxy
Server Stub	Skeleton	Stub	Service Implementation Template

With respect to the main transport protocols of Web services, the corresponding CORBA components are shown as in Table E.2.

Table E.2 Transport protocol components of CORBA and Web Services

	CORBA	Web Services
Protocol	IOP, GIOP	SOAP, HTTP, XML Schema
Location Identifiers	IORs, URLs	URLs
Interface spec	IDL	WSDL
Naming,Directory	Naming Service, Interface Repository, Trader service	UDDI

E.3.1 WSDL & IDL

Similar to CORBA IDL, The XML-based WSDL contains the abstract definition for a service, which defines both types and messages. WSDL also contains a concrete section that defines how the service is contacted, for example, protocol, encoding, and URI details.

E.3.2 IOP & SOAP

The CORBA IOP specification defines a very efficient binary protocol. SOAP is text-based and optionally includes type information as part of the message, which simplifies debugging and traffic monitoring because the message content is human-readable text. The CORBA IDL type system cannot accommodate certain requirements, such as DOC or PDF files as part of message. The SOAP with attachment specification allows MIME attachments to be included as part of the message content.

CORBA IDL is bound to IOP as a transport mechanism, whereas WSDL uses SOAP, which is not tied to any transport protocol. Typically, SOAP uses HTTP (Hypertext Transport Protocol) protocol, but commercial SOAP implementations already support other protocols such as HTTPS (Secure Hypertext Transport Protocol), SMTP (Simple Mail Transfer Protocol), and JMS (Java Message Service).

E.3.3 CORBA Services & UDDI

A UDDI registry closely corresponds to the CORBA Trader Service and CORBA Naming Service, yet there are plans for another of UDDI that resembles the CORBA Naming Service and will offer a simplified view of its data.

E.3.4 Developing Model

The mechanisms for generating client and server components for Web-Services are similar to that for CORBA, as illustrated in Figure E.2. When auto-generated client-side stubs (service proxy) are used for Web Services, the development processes and the code are virtually identical for Web Service and CORBA solutions. Typically one starts with an interface definition (WSDL) for the service. A client-side stub (service proxy) is auto-generated from this interface. On the server-side, the interface is processed to yield a base class for the implementation class of a service that must be written by the developer with respect to the Web services. In terms of CORBA, the interface is mapped into server skeletons that invoke the object implementation through the server program that is also written by the developer in addition to the object implementation.

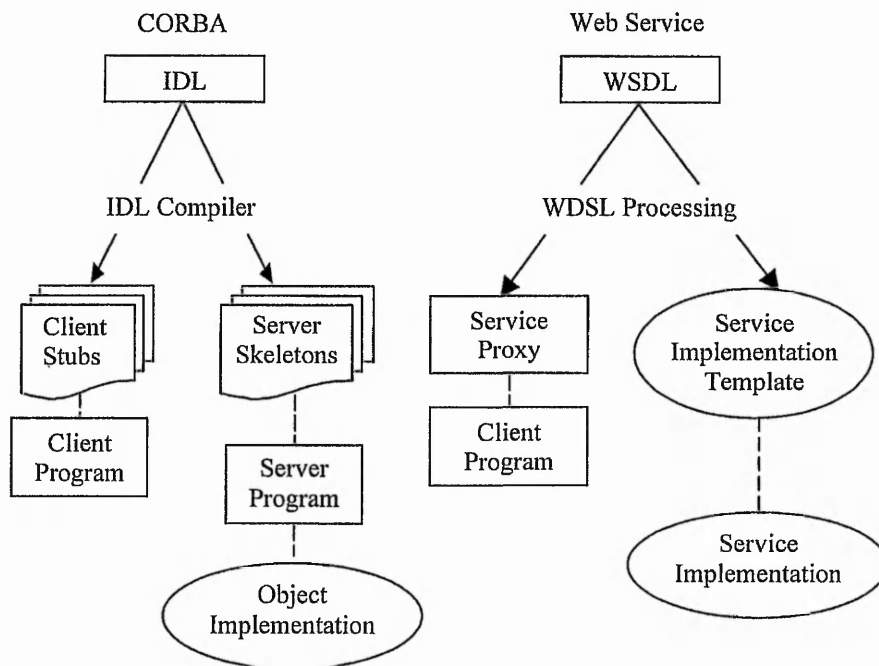


Figure E.2 Generation of client and server components from interface for Web Services and CORBA

E.3.5 Advantages and Disadvantages

CORBA, as a typical distributed application framework, has been around for a while and successfully applied to build complex services in a number of areas, ranging from telecommunications, finance, e-commerce, healthcare, to the graphical user interface

of Linux desktop (GNOME). There is, however, an unprecedented hype surrounding the new paradigm Web Services. In reality, the Web services paradigm lacks a precise definition. Furthermore, one of conclusion from some observations concerning CORBA and Web services is that whatever can be accomplished by CORBA can be accomplished using Web service technologies and vice versa, although the amount of efforts required would be noticeably different [117]. Nevertheless, there are still more attentions to try to apply the new programming paradigm Web services while many articles that evaluate and compares the features offered by each. In this section the main features of each surveyed in this research are provided and some possible applying models are presented.

Advantages of Web services:

- Web services provide interoperability between various software applications running on disparate platforms.
- Web services use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- By utilizing HTTP, Web services can work through many common firewall security measures without requiring changes to the firewall filtering rules.
- Web services easily allow software and services from different companies and locations to be combined easily to provide an integrated service.
- Web services allow the reuse of services and components within an infrastructure.

Disadvantages of Web services:

- Web services standards for features such as transactions are currently nonexistent or still in their infancy compared to more mature distributed computing such as CORBA.
- Web services suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing text-based formats. XML explicitly does not count among its design goals either conciseness of encoding or efficiency of parsing.

- By utilizing HTTP, Web services can evade existing firewall security measures whose rules are intended to block or audit communication between programs on either side of the firewall.

The main addresses Web services are used seem to be that they rely on HTTP over TCP (Transmission Control Protocol) port 80. Many enterprises have protected themselves by using firewalls that filter and block much Internet traffic for security reasons. In this environment, typically many (almost all) ports are closed to incoming and outgoing traffic, and the administrators of these firewalls are not eager to open them up. Port 80, however, is always open because it is used for Web browsers. Web services tunnel everything through port 80, making the technology very appealing.

Another main reason of utilising Web services is that they can provide a very loose coupling between an application that uses the Web service and the Web service itself. This should allow either piece to change without negatively affecting the other. This flexibility may become increasingly important as software is built by assembling individual components into a complete application.

The technical differences between CORBA and Web Services are mainly due to the different origins of them. CORBA focuses on a solution for industrial-strength applications within private or corporate networks. While Web services focus on lightweight, internet-based services, which can be reused and combined as required, decoupling clients from the service implementation. As much, Web services offer a great opportunity to reuse and extend CORBA systems.

According to the above comparisons between CORBA and Web Services, it can be concluded that there are some reasons to combine CORBA and Web Services to provide a more powerful distributed application. One of them is CORBA's high performance feature. In CORBA, there is a tight coupling between the client and the server. Both must share the same interface and must run an ORB at both ends. What's more, the interaction between client and server can be done directly, with no need for further intermediation (except from the ORB). In Web services, everything is

decoupled. The client sends a message and receives a message. The response does not give an immediate access to the next step. Web service implementations support different client-side application programmer interfaces.

From the view of engineering, there are many applications required to provide a high performance to make multiple applications work together. CORBA has been successfully used in engineering area in its high tight coupling and high performance features. The combination of CORBA objects and Web services can be one of best solution to provide a powerful application paradigm in distributed systems.

E.4 Building Web Services from CORBA

CORBA developers typically want to expose existing CORBA-based logic as one or more Web Services interfaces. This requires the IDL-to-WSDL tool to provide full support for complex data-types and user-defined data constructs. It is not feasible to modify the IDL, because this requires changes to the existing business logic. Cape Clear Studio development tool and Cape Clear Server runtime platform can be used to integrate Web Services with CORBA logic works in practice [118]. With a specific CORBA ORB provided in Cape Clear Studio and Cape Clear Server, new and existing components deployed in the CORBA ORB as Web Services, as depicted in Figure E.3. The development details can be found in [117].

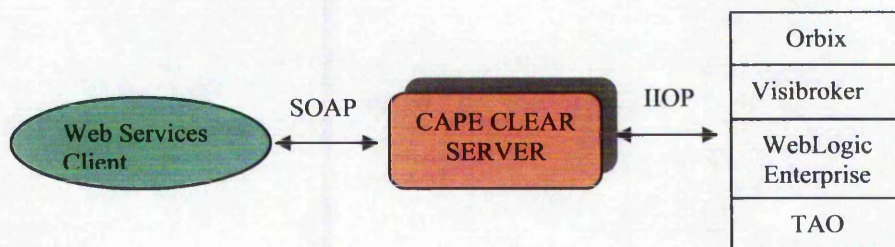


Figure E.3 Exposing deployed CORBA ORB components as Web Services

Optimistic visionaries predict a day when “millions of Web Services” are commercially available to businesses for use as the building blocks for modern software systems, and are used internally or externally. According to IDC (Industrial Development Corporation), a global market intelligence firm in the information

technology industry, approximately 3,300 Web services-based projects were implemented in North America in 2002 alone. IDC estimates that spending on Web services hardware, software, and services will reach \$15.2 billion by 2007 [119]. However, a more practical approach for an initial project involves the reuse of existing CORBA logic, along with Web Services technology to expose these systems in a new way. Web services can be used as middleware for middlewares such as CORBA.

The strategy exposing CORBA in Web services enables original CORBA-based applications deployed across firewalls: SOAP (over HTTP or HTTPS) can be used to integrate applications across firewalls. In this pattern, mainstream developers (such as JSP or Visual Basic programmers) develop the client-side application, while specialists CORBA programmers write the back-end logic.

CORBA is not only regarded as a legacy system to be wrapped into a Web service, but also can be used as a complement in nature with Web services. CORBA provides a mature middleware infrastructure, with robust and scalable features and services, for building mission-critical systems.

E.5 Intelligent Engineering Design Web Services

Collaborative product design may involve a number of software applications that run on geographically distributed computers. For example, designers, material specialists, manufacturing engineers, and structural analysts of a product may reside in different locations and use separate computer systems and software packages for design and analysis.

It is inevitable to build distributed infrastructure based on the Internet and Web to integrate collocated and distributed design resources for collaborative product design. The Web services model is becoming a potential approach for integrating business applications in that the model can improve the flexibility and extend the functionalities of a software application by making it interoperable with other software services.

The proposed framework is a collaboration model based on the Web service. The framework has three layers: user layer, design service broker layer, and design service layer, as shown as in Figure E.4.

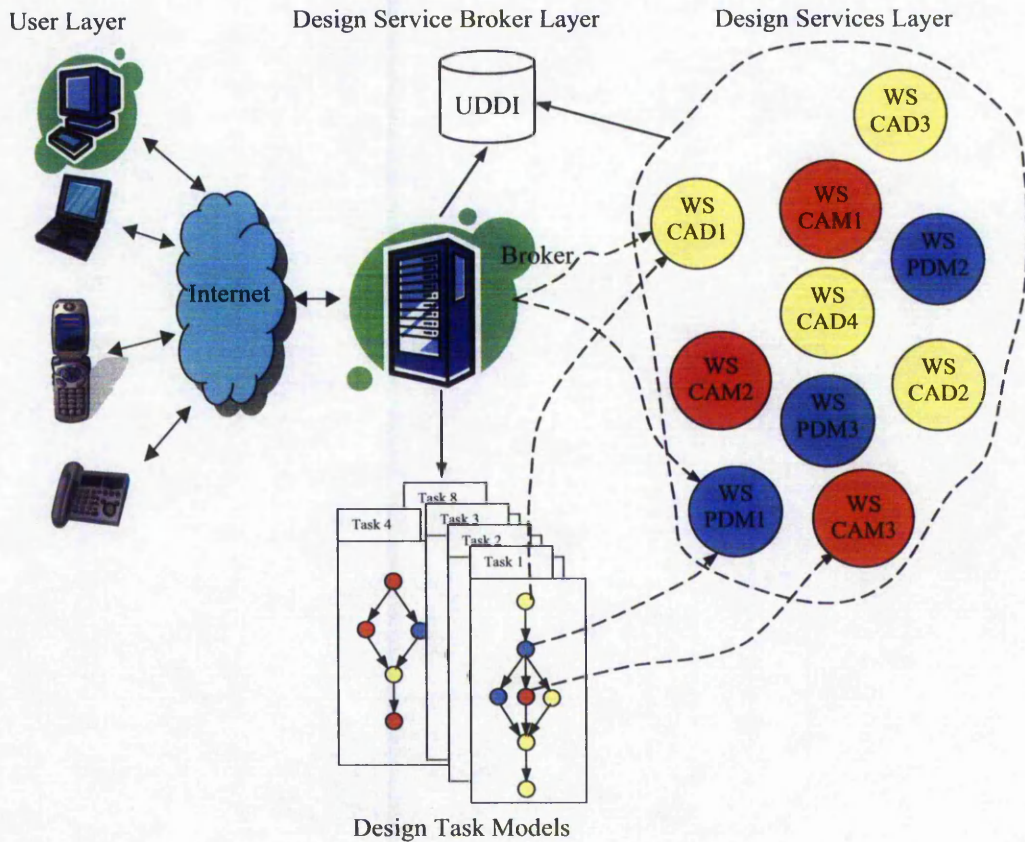


Figure E.4 Framework of intelligent engineering design Web services

The user layer provides public client interfaces for engineering designers to access the available services. Distributed and collocated engineering designers can access services from any Web-enabled device. The design service broker layer acts as a service broker, supporting service registration and publication. Each service is registered to the agency by publishing its service location, service type. Each service can be represented as a service component, and thus another process can call it. This enables any combination among multiple design activities to fulfil a complicated design task. The broker provides a design task model warehouse where each model of service represents a design process map with certain design objective. The service layer consists of engineering service components designed by Web services. These

services are published through UDDI and can be communicated with their legacy applications.

When a user invokes a design service via the public interface provided in a ubiquitous environment, a request is sent to the design service broker, which then performs searching and matching an appropriate design service model by looking up the services warehouse. When an appropriate task model is found, the broker binds each activity in the model to an appropriate design service available, calls a corresponding client-side interface and returns it to the user. From the client interface, the user could fulfil design via invoking remote services. Whether client program or each service is designed comply by the protocols, i.e. WSDL/UDDI/SOAP.

E.6 Developing Issues

E.6.1 Definition of Each Web Service

WSDL is used to define the interface of a Web service. It defines the syntax, the semantics, and all the various administrative aspects to a Web service procedure call. The service might be a new application, or even an existing application. Specifically WSDL provides a number of key pieces of information:

- A definition of the format of the messages that are passed between two endpoints using its <types> and <message> elements and appropriate schema definitions.
- The semantics of the service: how it might be called to make a synchronous request/reply, synchronous reply-only or asynchronously communicate.
- The end point and transport of the service via the <service> element: that is, who provides the service.
- An encoding via the <binding> element, that is how the service is accessed.

E.6.2 Intelligent Design Broker

In order to actually use a service, a user on client side must first find that service, retrieve information about how to use the service, and understand who might provide the service. The Universal Discover and Description and Integration specification, or

UDDI, defines a number of lookup services aimed at allowing clients to look up and retrieve the required information to access a Web service.

UDDI actually provides three specific services:

- Traditional white pages for looking up a Web service by name.
- Traditional yellow pages for looking up a Web service by topic.
- Green pages for more generic searches based on the characteristics of a Web Service.

In this application, a user request is sent to the service broker, which is also used as a client to forward it to UDDI. As an actual application model, the design broker is an intelligent agent to assess the user primary needs, processing the request, optimising the objectives and then provide an appropriate task model corresponding to the dynamic request. The task model includes the workflow map binding to the essential services. It might be a linkage to a service or multiple services. The design broker itself is represented as a Web service component and is attached to the task model that links to the function services at service layer.

Any aspects of design process control and management could be designed as a broker, for example, conversation broker, collaborative broker, dynamic binding broker, and so on. Flexible communication between the broker service and function services enables the implementation for any design intention corresponding to any combination of function services. To a developer of distributed environment, available Web services can be used as elements for constructing more complex applications. Application design is not only based on the code programming but also based on the functional software service integration.

E.7 Summary

Web services, as a new approach in distributed computing, aims to build flexible architecture to enable the communication over ubiquitous text-based protocols between disparate resources. More and more attentions have been drawn to this new distributed computing model.

CORBA provides high performance communication in its binary protocol even though it suffers from the limitation of ubiquitous environment. CORBA can be combined with Web services to obtain a more powerful distributed computing model.

The CORBA objects can be wrapped as Web service according to Web service specification, without modification of original code. This makes it easier to incorporate CORBA as a complement of Web services.

Comparing and contrasting between CORBA and Web services help CORBA developers to understand Web services and fast turn the CORBA object into a Web service. In an environment based on Web services, the CORBA objects wrapped into a service can be registered, searching, or binding according to the Web service specification.

The application paradigm is illustrated by a collaborative design environment based on the Web services over the Internet. The advanced intelligent broker is a mediator between service users and services. The user's design intention can be parsed, processed and transformed into a dynamic task model linking to essential services to fulfil the design task.