UNIVERSITY OF CALGARY

Distributed Database and Knowledge Base Modeling for Concurrent Design

by

Fengdong Zhang

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL AND MANUFACTURING ENGINEERING

CALGARY, ALBERTA

DECEMBER, 2000

©Fengdong Zhang 2000

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-65016-2

Canada

ABSTRACT

This research focuses on the development of a distributed database and knowledge base modeling approach and an Internet-based concurrent design system. Geographically distributed databases and knowledge bases, representing product development life-cycle activities, are integrated through the Internet. The consistency of the distributed databases is maintained using the distributed data dependency relation maintenance mechanism developed in this research. A distributed knowledge-based inference mechanism is introduced to create product life-cycle databases automatically. Based upon the distributed database and knowledge base modeling approach, a concurrent design system has been developed for modeling concurrent design alternatives. The optimal alternative is identified using either the exhaustive method or the Genetic Programming method. The optimal values of the design parameters are identified using the Particle Swarm Optimization method. The system has been implemented using VisualWorks. Example applications have been developed to illustrate the effectiveness of the concurrent design system.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my profound gratitude to my supervisor Dr. D. Xue for his guidance, encouragement and continuous support during my course of study and research in the Department of Mechanical and Manufacturing Engineering. Thanks also go to my examining committee members, Dr. D. H. Norrie and Dr. M. Ulieru, for their critical review of this thesis.

I would also like to thank the Faculty of Graduate Studies and the Department of Mechanical and Manufacturing Engineering for their generous financial support during my study at University of Calgary.

Thanks are extended to the supporting staff of the Department of Mechanical and Manufacturing Engineering, especially Nareeza Khan, Nick Vogt, Lynn Banach, Khee Teck Wong, and Dan Forre, for their help and support during the course of my graduate studies.

I am also grateful to my friends for what they have done in different ways to help and encourage me to complete my M.Sc. program.

Last but not least, my wife Dongmei Wang and my son Kan Zhang deserve my deep appreciation for giving me unconditional support at all times.

TABLE OF CONTENTS

APPROVAL PAGE ii

ABSTRACT iii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS v

CHAPTER 1: INTRODUCTION 1

1.1 Introduction 1

1.2 Problem Statements 4

1.3 Research Objectives 6

1.4 Overview of This Research 7

1.5 Organization of This Thesis 8

CHAPTER 2: RESEARCH BACKGROUND 11

2.1 Literature Review 11

2.1.1 Product Modeling 11

2.1.2 Product Development Techniques 14

2.1.3 Computer-Based Systems for Product Development 15

2.1.4 Integration of Distributed Product Databases and Knowledge Bases 17

2.1.5 Applications of Multi-Agent Systems in Product Development 18

2.1.6 Internet-Based Product Development 20

2.2 Related Techniques 22

2.2.1 Feature-Based Modeling 23

2.2.2 Concurrent Engineering 24

2.2.3 Object-Oriented Programming and Smalltalk 26

2.2.4 Distributed Systems and the Internet 27

2.2.5 Global Optimization 29
2.3 A Feature-Based Database and Knowledge Base Representation Scheme 30
2.3.1 Database Representation 30
2.3.1.1 Class Features and Instance Features 30
2.3.1.2 Maintenance of Data Dependency Relations 32
2.3.2 Knowledge Base Representation 33
2.3.2.1 Rule-Bases 33
2.3.2.2 Reasoning with Rule-Bases 34
CHAPTER 3: DISTRIBUTED PRODUCT DATABASE AND KNOWLEDGE BASE MODELING 36
3.1 Introduction 36
3.2 Distributed Product Database and Knowledge Base Modeling Architecture .. 37
3.2.1 Product Development Life-Cycle Activity Modeling 37
3.2.2 The Client-Server Communication Architecture 38
3.2.3 Architecture of the Distributed Database and Knowledge Base Modeling System 40
3.2.4 Node Definitions and Node Connections 43
3.2.4.1 Definition of Internet Nodes 43
3.2.4.2 Connection of Internet Nodes 45
3.3 Distributed Feature-Based Database Modeling 46
3.3.1 Virtual Features 46
3.3.1.1 Virtual Class Features 46
3.3.1.2 Virtual Instance Features 47
3.3.1.3 Generation of Instance Features from Virtual Class Features 49
3.3.2 Modeling Database Relations 51
3.3.2.1 Modeling Database Relations at Class Feature Level 52
3.3.2.2 Modeling Database Relations at Instance Feature Level 53

3.3.3 Maintenance of Dependency Relations among Distributed Data 53
3.3.3.1 An Algorithm for Maintaining Distributed Data Dependency Relations.. 55
3.3.3.2 An Example of Attribute Propagation Process 57
3.4 Distributed Knowledge Base Modeling 58
3.4.1 Virtual Rule-Bases 59
3.4.2 Selection of Virtual Rule-Bases 61
3.4.3 Reasoning with Distributed Rule-Bases 63
3.5 Summary 65
CHAPTER 4: CONCURRENT DESIGN BASED UPON DISTRIBUTED DATABASE AND KNOWLEDGE BASE MODELING 67
4.1 Introduction 67
4.2 Modeling of Product Realization Process Alternatives 69
4.2.1 The Relations among Internet Nodes 69
4.2.1.1 Logical Relations among Internet Nodes 70
4.2.1.2 Creation of Internet Node Relations 71
4.2.2 Representation of Product Realization Process Alternatives 72
4.2.2.1 Product Realization Process Alternatives 72
4.2.2.2 Display of Product Realization Process Alternatives 73
4.3 Identification of the Optimal Product Realization Process Alternative 75
4.3.1 The Exhaustive Method 76
4.3.1.1 The Algorithm for Generating All Alternatives 76
4.3.1.2 An Example of Generating All Alternatives 76
4.3.2 The Genetic Programming (GP) Method 78
4.3.2.1 Introduction to Genetic Programming Method 78
4.3.2.2 Genetic Programming for Alternative Optimization 82
4.4 Identification of Optimal Design Parameter Values 90
4.4.1 Introduction to Particle Swarm Optimization (PSO) 90

4.4.2 PSO in Design Parameter Optimization 92
4.4.2.1 Formulation of Parameter Optimization Problems 93
4.4.2.2 Issues of Parameter Optimization with PSO 94
4.4.2.3 The Parameter Optimization Interface 95
4.5 Summary 96
CHAPTER 5: SYSTEM IMPLEMENTATION AND APPLICATION EXAMPLES 98
5.1 System Implementation 98
5.1.1 System Interfaces 98
5.1.2 New Classes Developed for System Implementation 101
5.1.3 Message Handling 103
5.2 Application Examples 104
5.2.1 The Concurrent Design Problem 105
5.2.2 Generation of Instance Features 107
5.2.3 Rule-Based Reasoning with Virtual Rule-Bases 107
5.2.4 Propagation of Changed Attribute Values 110
5.2.5 The Optimization of Design Parameter Values Using PSO 111
5.2.6 The Optimization of Product Realization Process Alternatives Using GP 114
CHAPTER 6: CONCLUSIONS AND FUTURE WORK 120
6.1 Conclusions 120
6.1.1 Distributed Database and Knowledge Base Modeling 120
6.1.2 Internet-Based Concurrent Design 122
6.2 Future Work 124
REFERENCES 127

CHAPTER 1

INTRODUCTION

In this chapter, a brief background introduction for this thesis is provided. The problems remaining in distributed database and knowledge base modeling for product development are summarized. Based on these problems, the objectives of this research are outlined. The organization structure of the thesis is given at the end of this chapter.

1.1 Introduction

Development of a product is carried out through a sequence of processes including marketing, design, process planning, manufacturing, etc. With the advances of computer technologies and information technologies in the new century, global competition is becoming the main characteristic of the marketplace. In responding to the increasingly dynamic market requirements on product development time, cost, and environmental concerns, etc., many methodologies and computer-aided systems have been developed and used to improve the overall performance of products.

Among product development activities, design is the major process that affects the performance of other down-stream life-cycle phases. An effective design should be identified based on customer requirements, with consideration of the down-stream performance of the design. Methodologies for improving product life-cycle performance include Design for Manufacturing (DFM) [Helander and Nagamachi 1992], Design for Assembly (DFA) [Magrab 1997], and Design for Environment (DFE) [Magrab 1997]. Design for Manufacturing (DFM) is an approach that incorporates manufacturing process information into the design process. The basic objective of this approach is to reduce manufacturing costs and lead-time by considering manufacturability aspects of the product at the design stage. In Design for Manufacturing, information flows between the process models of design and manufacturing. Similarly, Design for Assembly (DFA) and Design for Environment (DFE) emphasize the assembly performance and environmental impact of the product design respectively [Magrab 1997].

To evaluate down-stream product life-cycle performance and use evaluation measures to improve design, the concept of concurrent engineering was proposed [Parsaei and Sullivan 1993]. Many studies in concurrent engineering focus on incorporating all relevant down-stream life-cycle aspects into the design stage. In these studies, the activities in down-stream product development phases, such as production process planning, manufacturing, service, and recycling, are handled concurrently with the design process. However, the process of concurrent design is very complex, since it involves dynamic information flows among all related product development processes. Therefore, the concurrent design methods and tools that can be used effectively to improve product development life-cycle performance must be investigated.

To improve product development efficiency and apply the methodologies mentioned above, many computer-based systems have been developed [Singh 1995]. Computer-Aided Design (CAD) systems help designers produce modifiable product designs easily and perform design-related analysis efficiently. The efficiency and productivity of the design process are greatly improved using CAD systems. Computer-Aided Process Planning (CAPP) and Computer-Aided Manufacturing (CAM) are used to assist process planning and manufacturing activities respectively. Among these computer-aided systems, Computer-Aided Design (CAD) is widely used for product modeling by 2D and 3D geometry as well as in animation. CAD databases can be regarded as product geometric models and can be used to generate down-stream product development data such as manufacturing processes and assembly processes.

Product design is a complex process that involves considerable knowledge and decision-making. Techniques in Artificial Intelligence (AI), such as expert system, fuzzy logic, neural networks, genetic algorithm, etc., have been applied to computer systems to improve product development capabilities and efficiency.

Of all the issues related to computer-aided product development, the modeling of product databases and knowledge bases is one of the most important. Feature-based product modeling is one of the approaches used in computer-aided systems [Gardan and Minich 1993, Xue and Dong 1993]. The concept of feature was originally used for

representing geometric primitives, such as blocks and holes, for modeling product geometry [Shah and Mantyla 1995]. In Xue's previous work, the concept of feature was extended to model other product life-cycle primitives, including design primitives and manufacturing primitives, for improving product life-cycle modeling efficiency [Xue and Dong 1993, Xue et al. 1999]. Unlike CAD systems in which mainly geometric information of a product is modeled, the feature-based product life-cycle modeling approach can build product descriptions in a more natural way. It uses not only the geometric primitives, but also non-geometric properties and the relations among these properties.

Object-oriented databases model products more efficiently than relational databases because of the properties of inheritance, encapsulation, etc. Thus, a product family can easily be modeled by a class and its sub-classes.

In most presently developed computer-based product development systems, the databases and knowledge bases are modeled at the same location. In a concurrent design system, all down-stream aspects of product development, such as marketing, design, manufacturing, and service, must be considered concurrently; however these activities, usually take place in different geographical locations. It is very difficult to implement concurrent design without the assistance of an effective computer system that can handle distributed databases and knowledge bases.

Current distributed modeling and computing techniques focus on integrating the objects developed on different platforms (such as UNIX, MS Windows) using different computer languages (such as C, C++, Java) into the same environment. Typical distributed object modeling methods include Distributed Component Object Model (DCOM) [Grimes 1997], Common Object Request Broker Architecture (CORBA) [Otte et al. 1996], and Remote Method Invocation (RMI) [McCarty and Cassady-Dorion 1999]. These methods provide interoperability between applications on different machines in heterogeneous distributed environments.

The recent development of multi-agent systems provides another approach for associating separated databases and knowledge bases and their related modeling systems

[Norrie 1999]. Multi-agent systems are software societies that handle all related tasks through communication, negotiation, collaboration and other activities among relevant agents. In order for a multi-agent system to be successful, the agents must be highly intelligent.

1.2 Problem Statements

Despite considerable progress in computer-aided systems for assisting product design and manufacturing, problems still remain in modeling distributed databases and knowledge bases for concurrent engineering design. These problems are summarized as follows.

1). Problems in distributed database modeling

- In distributed object modeling approaches including DCOM, CORBA, and RMI, association of the objects at different locations is predefined by compiled computer programs. However in the product development process using concurrent design methodology, product realization process alternatives are generated and evaluated dynamically in order to identify the best solution. The distributed databases must be associated in a dynamic manner.
- Distributed object models are primarily used for representing objects, which are described by data and functions to access these data. Collaboration of these objects is limited to only the data and functions. Since the computer-based product development process involves more sophisticated descriptions such as composing elements, qualitative relations, dependency relations, and constraints, modeling of these descriptions is also needed. These descriptions are usually associated with distributed databases.

2). Problems in distributed knowledge base modeling

- Multi-agent systems aim at decomposing a problem into many sub-problems and using the knowledge of different agents to solve the different types of sub-problems. The collaboration of agents is usually conducted through brokers called mediators. In

a multi-agent system, an agent is associated with certain knowledge for solving problems of a certain type. In the product development process, since different knowledge bases, including design knowledge bases and manufacturing knowledge bases, are required to access the same database, a more flexible mechanism is required to dynamically select and combine knowledge bases at different locations.

- In existing distributed systems, databases and knowledge bases are not well integrated. Databases are usually modeled as objects or relational databases. Knowledge bases are usually described by IF-THEN rules. The feature-based product modeling system provides a new approach for integrating the databases and knowledge bases. It introduces features that are described by both qualitative descriptions for symbolic reasoning and quantitative descriptions for numerical calculation [Xue and Dong 1993, Xue et al. 1999]. In existing knowledge-based concurrent design systems, rule-based reasoning is conducted only at one location. To incorporate distributed databases and knowledge bases for concurrent design, a distributed inference mechanism must be developed.

3). Problems in concurrent design with distributed databases and knowledge bases

- Concurrent design has been recognized as a method that can lead to lower production costs, less lead-time, and better life-cycle performance. It has been widely accepted as an engineering philosophy. Many computer-based tools have been employed for modeling concurrent design [Reidsema and Szczerbicki 1997]. Most of these tools, however, can handle only centralized databases and knowledge bases. A concurrent design tool that employs distributed product databases and knowledge bases will increase the advantages of a concurrent design system.

- There is still little information on how a concurrent design candidate can be modeled, evaluated, and modified dynamically in terms of using a distributed database and knowledge base modeling system.

1.3 Research Objectives

In order to address the problems stated in Section 1.2, the objectives of this research are summarized as follows:

1). To develop a distributed database and knowledge base modeling system with the functions required for concurrent engineering design. These functions are:

- The distributed database and knowledge base modeling system is open and dynamic. The product modeling databases and knowledge bases, representing product development activities, are associated by the Internet. Each database model can be included in or excluded from the system by connecting the model to the system and disconnecting it from the system. This function provides choices of available databases for concurrent design.
- Product descriptions at different locations are kept consistent all the time by a relation maintenance mechanism. This mechanism is the engine that propagates any data changes to all related data, no matter where these data are located. It is critical to the success of the system in concurrent design, since any value changes in design databases will lead to changes in all down-stream product development modeling databases.
- Distributed knowledge bases should be integrated and used to access the same databases. By means of such integration, knowledge at remote locations can be used to assist local product modeling processes. To accommodate rule-based reasoning into the distributed concurrent design system, a distributed inference mechanism must be developed to conduct product development activities concurrently.

2). A concurrent design system must be developed, based on the distributed database and knowledge base modeling approach. This system should have the following functions:

- Alternative product realization processes can be modeled effectively so that the concurrent design system can generate and evaluate the alternatives.

- The optimal product realization process is reached through optimization conducted at two different levels. Parameter optimization is employed to identify the optimal parameter values for an alternative product realization process. Based on the results of parameter optimization, alternative optimization is carried out for identifying the optimal product realization process. Through these processes, all relevant product development activities interact dynamically to reach the goal of concurrent design.

1.4 Overview of This Research

This research is composed of two parts: development of the distributed database and knowledge base modeling system, and application of this system in concurrent design. The distributed database and knowledge base modeling system can be used to model and integrate product development life-cycle activities that are geographically distributed. The concurrent design system is used to identify the optimal product realization process alternative based on the distributed database and knowledge base modeling approach.

In the distributed database and knowledge base modeling system, different product development activities are modeled in different computers at different locations. All the computers are connected to the Internet. Each computer is represented as an Internet node. The communications among these nodes are conducted through messages over the Internet.

Distributed database modeling is the core part of the distributed database and knowledge base modeling system. The product databases, i.e., the product life-cycle aspect models, are built using primitives called features. Features are described at two different levels, class level and instance level. Features preserved in remote nodes are called virtual features. Virtual class features can be used to generate true instance features at the local node. Virtual instance features are actual product modeling databases preserved in different remote nodes. These product data can be accessed from the local node. The relations among virtual instance features and true instance features can also be defined to establish relations of the related databases. The relations can be modeled at both class feature level and instance feature level. Through these relations, data changes

in one node can be propagated to other related nodes automatically by a relation dependency maintenance mechanism. This function is critical to implementing concurrent design methods.

In knowledge base modeling, knowledge is represented by rules which are organized into separated rule-bases. The rule-bases preserved in remote accessible nodes are called virtual rule-bases. Virtual rule-bases can be selected for reasoning at the local node together with true rule-bases. To assist in product development, each instance feature can be associated with a number of rule-bases including virtual rule-bases. The product databases can be generated and modified through rule-based reasoning. Since the product databases are distributed at different locations, a distributed inference mechanism is developed to automatically activate inference processes in remote nodes.

Based on the distributed database and knowledge base modeling approach, a concurrent design system was developed. In this system, a product realization process alternative is modeled by a collection of Internet node names that represent different product development activities such as design, manufacturing, and assembly. If the number of alternatives is small, the alternatives can be generated using an exhaustive method. When the number of alternatives is large, the optimal alternative can be identified through a two-level optimization process, parameter optimization and alternative optimization. Parameter optimization is conducted to obtain optimal values of parameters for the alternatives. Particle Swarm Optimization (PSO) [Kennedy and Eberhart 1995], a global optimization method, is used for parameter optimization. Based on the results of parameter optimization, alternative optimization is conducted at the second level to identify the optimal alternative. The Genetic Programming method (GP) [Koza 1992, Angeline 1994] is used for alternative optimization.

1.5 Organization of This Thesis

There are seven chapters in this thesis. Chapter 2 starts with a detailed literature review that provides the research background of this work. This review covers topics related to the work presented in this thesis, including design-for-X techniques,

concurrent design methods considering all relevant life-cycle aspects, product modeling approaches, distributed systems for concurrent engineering design, and Internet-based applications in concurrent design. The existing techniques that were used in this research, including feature-based modeling, object-oriented programming, distributed systems, the Internet, and engineering optimization, are then briefly introduced. Since this research is based on a previously developed system – a feature-based intelligent design system [Xue et al. 1999], the structure and key functions of this system are described in this chapter.

Chapter 3 gives a detailed description of the distributed database and knowledge base modeling system. First, the architecture used for integrating geographically distributed databases and knowledge bases through the Internet is introduced. Then details of distributed database modeling and distributed knowledge base modeling are described. As one of the core functions needed for concurrent design, distributed database modeling is emphasized in this chapter.

Starting from the basic definitions of virtual features, including virtual class features and virtual instance features, the modeling techniques of virtual features and data relations involving virtual features are presented in detail. A mechanism for maintaining data dependency relation is then discussed by introducing an automated data propagation algorithm and an example.

For knowledge base modeling, the virtual rule-base concept is introduced. Techniques for using virtual rule-bases are described. A distributed reasoning algorithm is then discussed.

Chapter 4 presents a concurrent design system. This system is based on the distributed database and knowledge base modeling approach described in Chapter 3. The concurrent design system provides the functions of modeling product realization process alternatives and generating the optimal concurrent design solution. Two levels of optimizations are employed in this system. First, the design parameter values are optimized by Particle Swarm Optimization (PSO). Then the product realization process alternatives are optimized using Genetic Programming method (GP). These methods are described in detail in this chapter.

Chapter 5 discusses the issues in the implementation of the distributed database and knowledge base modeling approach and the concurrent design system. These issues include data structures, class definitions and message handling.

Chapter 6 presents examples to illustrate the application of the distributed database and knowledge base modeling approach and the concurrent design system.

Chapter 7 summarizes this thesis and gives conclusions. Possible future research directions are also discussed in this chapter.

CHAPTER 2

RESEARCH BACKGROUND

This chapter presents a general review on subjects relevant to the research introduced in this thesis. Subjects include product modeling and development, computer-based systems for product development, and distributed modeling techniques for product development. As the basis of this research, the previously developed feature-based intelligent design system [Xue et al. 1999] is introduced. Techniques used in this research are also briefly described.

2.1 Literature Review

Product development is a complex process that involves human intelligence and available techniques. To improve the efficiency of product development and the performance of product life-cycles, many product modeling techniques and computer systems have been developed. Recent advances in computing technology and the Internet provide new approaches for developing and implementing more robust computer-based systems to assist product development. Such systems can play an important role in satisfying the requirements of the global market today.

2.1.1 Product Modeling

In computer-based product development, the techniques of product modeling are important in improving the effectiveness of product development systems. CAD systems have been widely used for modeling product geometry [Singh 1995]. Three types of geometric models are usually used for representing a product. They are the wireframe model, the surface model, and the solid model. The wireframe model builds a product using its boundary lines and curves [Lee 1985]. This model is simple but ambiguous in geometry interpretation. The surface model describes a geometric model using boundary surfaces, such as plane surfaces, surfaces of revolution, etc. [Mortenson 1985]. The surface model visualizes products better than the wireframe model. The solid model

provides more information of product geometry including topological relations among geometric elements. Primitives such as spheres, cylinders, cones, blocks, etc. are used to build a solid model [Tan et al. 1987]. The solid model takes much more computer memory than the other two models, but is the most suitable for product geometric modeling with a computer-based system.

The solid model has been used by conventional CAD systems for modeling product geometric information. However, modeling of product geometry is only one aspect of product modeling. It should be possible to model more information, such as generic relations among related products, non-geometric information, etc. For such purposes, the feature model has drawn the attentions of researchers [Shah and Rogers 1988, Xue and Dong 1993, Gardan and Minch 1993, Shah and Mantyla 1995]. A feature is a description necessary for modeling one aspect of a product. The feature model catches not only geometry information, but also non-geometry information of products. More details about feature-based modeling are introduced in Sections 2.2 and 2.3 of this chapter. Features of a product can be classified into different categories. Examples are material features, manufacturing features, technological features, and geometric features [Shah and Rogers 1988, Vickers and Swanson 1988, Shah 1989]. In feature-based product modeling research area, feature recognition is an approach for extracting the geometry from the CAD database for planning production processes [Henderson 1984]. Design-by-features is another approach for modeling a product using manufacturing features at the very beginning [Shah and Rogers 1988]. An international product modeling standard, STEP, has been developed to integrate the different product life-cycle models, using a universal computer language [Gu and Chan 1995].

Research on functional modeling has also been reported [Nagamatsu et al. 1999]. This research showed that the functional model has the potential capability of providing dynamic functional performance of products. It is suggested that the functional model be composed of block diagrams for explaining the functions, and mathematical models for simulations. This method is still in the early developing phase.

Product database modeling is one of the important issues in product modeling,

especially for computer-based modeling systems [Waldron et al. 1992, Shah and Mantyla 1995].

Conventionally, relational data models are widely used because they are easy to learn and easy to use. However, engineering product data are complex in terms of relations among the element entities. The relational data model has difficulty in modeling such relations [Seilonen 1995].

Object-oriented data modeling is a data modeling methodology proposed as an alternative to the relational modeling technique [Hughes 1991]. The object-oriented data modeling technique leads to more maintainable and understandable models that correspond more closely to real world entities [Lee and Sen 1994]. The application of the object-oriented database modeling in product development has been proven effective [Xue and Dong 1993, Yadav 1999]. The object-oriented database modeling technique is often combined with feature-based product modeling technique to improve the effectiveness of product modeling [Xue and Dong 1993, Yadav 1999].

In some cases, knowledge-based techniques are introduced to help manage the databases. Stonebraker proposed an active database management system called active DBMS [Stonebraker 1992]. Active DBMSs are databases that automatically carry out triggered actions when certain situations arise. The active behaviors are specified by production rules integrated in the system.

In the work of Bassiliades and Vlahavas [1997], a rule integration scheme in an object-oriented database management system (OODBMS) was presented. An active knowledge base system called DEVICE resulted from this work. Domazet and San [1997] described a system that integrates an expert system with a passive object-oriented management system to create active behavior in a single workspace environment. This system is regarded as an active database server. Roller and Eck [1999] presented an approach to a shared knowledge base for product development called the Active Semantic Network (ASN). The ASN is an intelligent knowledge base that adapts conventional database functions to the particular requirements of modern cooperative product design.

2.1.2 Product Development Techniques

Product development involves activities such as marketing, design, manufacturing, and service. Among these activities, design is the primary process that affects the performance of the product in all down-stream life-cycle aspects. Eighty percent of manufacturing decisions result directly from the design stage [Vliet et al. 1999]. In order to improve the competitiveness of products, a number of product development techniques that concurrently consider down-stream aspects of the product development process have been developed.

Design for Manufacturing (sometimes called Design for Manufacturability) [Helander and Nagamachi 1992, Magrab 1997, Vliet, et al. 1999] is widely accepted as an approach for creating product designs that eases the manufacturing task and reduces manufacturing costs. Conventionally, designers must be provided with up-to-date knowledge of manufacturing processes, tools and fixtures in order to improve the efficiency of the product realization process. Since manufacturing processes are complex, designers often have difficulty in fully considering all the requirements of manufacturability. That is where the DFM systems are placed. To emphasize different aspects of product development such as assembly, service, and environment at the design stage, techniques of Design for Assembly (DFA), Design for Serviceability (DFS), and Design for Environment (DFE) have also been developed [Magrab 1997].

Sharing many similarities with Design for "X" in terms of the concepts and objectives, concurrent design (or concurrent engineering) has been recognized as an approach to improve the quality and efficiency of product development. Concurrent design refers to the simultaneous design of a product and all its related processes in a manufacturing system, as well as related processes in later phases of the product's life-cycle [Parsaei and Sullivan 1993]. This means that all information flows should be multi-directional among the design processes and all related processes. Since 1980s, the benefits provided by concurrent design philosophy have been recognized and many industrial applications have been developed [Pennell et al. 1989].

There are two approaches to implementing the concurrent design practice: the team-based approach and the computer-based approach [Parsaei and Sullivan 1993]. The team-based approach is human-oriented. The members of the team are from all related functional areas. They can therefore contribute to the design of products and processes by identifying potential problems early and avoiding a series of costly reworks [Pennell et al. 1989].

Though it is easy to implement, this approach has apparent shortcomings: the difficulty and cost of managing the team, and team members' limited knowledge. So a team-based approach also needs the assistance of computer systems to enhance the team's performance. The computer-based approach is effective in integrating all related process models of product development into the same environment. With the increasing ability of handling large amount of information at high speed, computer-based systems are playing increasingly more important roles in implementing concurrent design.

2.1.3 Computer-Based Systems for Product Development

Among computer-based systems in product development, Computer-Aided Design (CAD) is one of the most widely used tools currently available to the industries. Conventional CAD systems are mainly used for geometric modeling and related computation and analysis [Singh 1995]. Even though it has been very successful in assisting designers to produce drawings and graphics in a fast and accurate way, CAD systems still have difficulties in handling non-geometric information about products. Computer-Aided Process Planning (CAPP) and Computer-Aided Manufacturing (CAM) are systems to automate process planning and other manufacturing activities [Singh 1995].

With the development of Artificial Intelligence (AI) techniques, knowledge-based systems have been used to improve the efficiency of product development and the performance of the product life-cycle [Court 1998, Judson et al. 1999]. Expert system is one of the techniques often used in product development. To improve the performance of conventional CAD systems, research has been conducted to introduce knowledge-based

systems to the CAD systems [Yoshikawa 1988, Anderson and Crawford 1988, Penoyer et al. 2000]. Penoyer et al. believe that future CAD systems should be open to integrating knowledge-based systems for all aspects of the product life-cycle [Penoyer et al. 2000]. Knowledge-based systems are also used in other computer-based systems to assist in manufacturing, assembly, etc. [Kroll et al. 1989, Colton 1993].

Recently, more research on integrating knowledge in product development has been done. Court [1998] identified important issues to be considered in order for knowledge or information to be successfully integrated into future product development. These issues include the media in which information and knowledge are provided, the manner in which information and knowledge are presented, and the location and administration of knowledge and information. Xue et al. [1999] presented a method for integrating knowledge bases with feature-based product databases for intelligent concurrent design. In Xue's previous research, an integrated and intelligent system was developed for modeling the databases and knowledge bases used at different product development phases [Xue and Dong 1993, Xue and Dong 1994, Xue et al. 1996, Xue 1997, Xue and Dong 1997, Xue et al. 1999]. In this research, product life-cycle aspects are modeled by aspect primitives called features, including design features such as mechanisms and components, manufacturing features such as holes and slots, and so on [Xue and Dong 1993]. A rule-based system was developed to generate product life-cycle aspect models automatically through rule-based reasoning [Xue and Dong 1994]. An optimization model was introduced to identify optimal design considering both functional performance and production cost [Xue et al. 1996]. The optimization model was improved based on genetic algorithm and simulated annealing to identify the optimal product realization process alternative and its parameter values [Xue 1997]. A design feature coding system and a manufacturing feature coding system were developed to organize large feature libraries and identify appropriate features during the product development process [Xue and Dong 1997]. Judson et al. [1999] discussed challenging issues of introducing knowledge-based engineering into an interconnected product development process. Even though a Design Structure Matrix (DSM) model was proposed to map the knowledge that

might be involved at the system interaction level for several components, more dynamic methods are still needed for such knowledge-based applications.

2.1.4 Integration of Distributed Product Databases and Knowledge Bases

In most of the presently developed computer-based product development systems, the product databases and knowledge bases are modeled at the same location. However product development activities, such as marketing, design, manufacturing, and service, usually take place at different locations. To take advantage of globally available product development resources, integration of these separated product modeling databases and knowledge bases becomes necessary in product development.

The research of distributed database modeling often focuses on integrating the objects developed on different platforms (such as MS Windows, UNIX, and Macintosh) using different computer languages (such as C++, Java, and Visual BASIC) into the same environment. In order to manage distributed databases, integration of distributed database management systems has been studied [Ozsu et al. 1994].

Research in this area also includes works on methodologies of collaboration and coordination of distributed product information management systems and frameworks that help designers make decisions. The SHARE project developed by Cutkosky et al. [1993] allows designers to gather, organize, and communicate design information over computer networks to establish shared understanding of the design. Groupware techniques are used in the SHARE project. Sriram and Logcher [1993] developed a computer-based design system that provides a shared workspace where multiple designers work in separate engineering disciplines. A global control system is used to solve problems of coordination and communication. A system proposed by Bliznakov et al. [1995] allows a designer to indicate the status of the tasks assigned so that other designers can follow over the computer network. This system incorporates a hybrid model for design information representation. Adamides [1995] presented a distributed active-resource coordination framework for a class of flexible manufacturing systems. Cooperative behavior is achieved by resolving conflicts and by maximizing the use of the

system's resources. This framework relies on a timed Petri net representation of the production responsibilities of each active resource in the system. Pahng et al. [1998] proposed a framework for modeling and evaluating product design problems in a computer network-oriented design environment. Design problems are decomposed into modules (such as a cost module) that represent different aspects of the problems. The modules can be distributed. A module can provide services to other modules through standard communication protocol.

Research of distributed knowledge base modeling has been conducted for developing multi-agent systems to solve problems through collaboration of different agents with different types of knowledge [Huhns and Singh 1998].

Some commonly used methods that provide interoperability between applications on different machines in heterogeneous distributed environment have become industrial standards. Typical distributed object modeling methods include Distributed Component Object Model (DCOM) [Grimes 1997], Common Object Request Broker Architecture (CORBA) [Otte et al. 1996], and Remote Method Invocation (RMI) [McCarty and Cassady-Dorion 1999]. These different distributed object modeling methods have been compared [McCarty and Cassady-Dorion 1999].

2.1.5 Applications of Multi-Agent Systems in Product Development

Multi-agent systems are another approach that can be used for product development in both centralized and distributed product development environments. Multi-Agent Systems, also called Intelligent Agent Systems, are software systems that are composed of program modules with intelligence and autonomy. These modules, regarded as agents, may collaborate dynamically to achieve the objectives of the systems [Norrie 1999].

Many applications have been developed using the multi-agent system approach for solving engineering problems [Shen and Norrie 1999]. Reidsema and Szczerbicki [1997] considered the complexity of implementing concurrent design that involves different life-cycle aspects of product development. They suggested that multi-agent distributed systems should be used as the core concept in developing a concurrent engineering

design system. Complicated concurrent design problems can be decomposed into subtasks that are distributed among different agents with the abilities to solve these problems. Coordination and cooperation among agents help achieve the goals of concurrent design, such as minimizing lead time, reducing manufacturing costs, and ensuring longer product life span.

An experimental multi-agent environment for engineering design was introduced by Shen and Barthes [1995] using techniques of distributed artificial intelligence. In this system, various design activities are modeled by a population of asynchronous cognitive agents. The agents communicate through a local network or the Internet. All agents in the system are autonomous and independent. Users of the system are regarded as human agents who are integrated into the design environment.

Danesh and Jin [1999] introduced an agent-based decision network framework for concurrent design and manufacturing. The design process is modeled using a decision-based approach. There are two major models in this framework: the decision-based design process model and the condition-based negotiation model. These models are introduced to help team members consider other members' decisions when making their own. Coherent design decisions among designers can therefore be achieved by explicitly representing and capturing individual design decisions and negotiation processes. Each designer is associated with an agent that is facilitated with the two models. This framework was found to be effective in integrating design and manufacturing processes.

Also for collaborative product development, the mechanism of agent-based workflow management was proposed to facilitate the team working in a collaborative product development framework [Huang et al. 2000]. In this framework, a web-based decision support system is used by team members who are geographically distributed. Agents are representatives of their human users. Each agent is assumed to be responsible for one work activity of the project. A limitation of the framework is that only static dependency relations, such as predefined predecessors and successors between agents, can be used in this system. Agent-based applications for product development also include information integration and collaborative service support in all aspects of a

product life-cycle [Gadh and Sonthi 1998, Tso et al. 1999].

2.1.6 Internet-Based Product Development

Recent advances in Internet technology provide new approaches for integrating the separated databases and knowledge bases into the same environment [Alles and Vergottini 1997, Huhns and Singh 1998]. Intensive research has been carried out on Internet-based or web-based product development. Name and Engelstein [1998] provide a brief overview of tools that could possibly be used to bring concurrent engineering to fruition. These tools include Email, Web sites, VRML (Virtual Reality Modeling Language), FTP (File Transfer Protocol), Multimedia, and Groupware.

Mendel [1999] predicted that the product data management software business would be reshaped by the Internet technology. Roy et al. [1997] reported a prototype framework of web-based collaborative product development. In this framework, all designers involved can collaborate through shared web pages and VRML models. Product modeling databases and the VRML-based geometric models are associated with shared web pages. Designers can access these data through hyperlinks. Adapalli and Addepalli [1997] described different ways of integrating manufacturing process simulations by means of the world wide web. Techniques used in this research include HTTP/CGI, java sockets, etc. It was concluded that performing manufacturing process simulations over the web is possible, even though some problems, such as the immaturity of related techniques, remain to be solved.

Methods for transmitting and viewing CAD data and engineering information through the Internet must be studied in order to develop web-based applications for product development. With this concern in mind, Kim et al. [1998] discussed the possibility of storing STEP data using the Virtual Reality Modeling Language (VRML) so that the product can be viewed in interactive 3D on a number of platforms using the Internet and World Wide Web. Formalisms for storing STEP data in an object-oriented database schema and converting STEP data to VRML are described. The prototype system, called CyberView, can provide support for members of distributed concurrent

engineering teams to share and exchange 3D information. For the purpose of developing a web–based DFX (Design for X) shell that is intended to be used to develop DFX tools, Huang et al. [1999] studied the technique of web-based product and process data modeling. A method called bills-of-materials is used for outlining product structures. A bill-of-materials is a list of the items or materials needed to produce a parent item. This method cannot model relationships between components and parts of the products effectively.

To support product modeling and collaborative design activities, Lee et al. [1999] presented an approach for web-enabled feature-based modeling in a distributed design environment. In this approach, there is a neutral feature model in the server. This model provides a generic naming scheme for naming consistency, so that the relationship between geometric entities of the server and clients can be maintained.

In product development, the life-cycle support of products can be enhanced by the Enterprise-Web portal in terms of information and resources sharing and management, according to Rezayat [2000a]. Rezayat also discussed problems that the web-based technology, especially XML (eXtensible Markup Language), is used for defining interfaces supporting knowledge capturing, storing, and sharing through out the product development life-cycle [Rezayat 2000b].

Another interesting prototype system called WebCAD has been developed to allow designers to define the geometry of parts [Kim et al. 1999]. The basic objective of this project is to provide manufacturing services, especially machining processes, through this tool over the Internet. In other words, this is a design interface that produces a high probability of success in respect to manufacturability of the design.

Based on the similar ideas, Higgins and Langrana [1999] developed a web-based user-friendly virtual design and fabrication system using the knowledge-based approach. Web technology is also used for communication and sharing information among designers during product development [Ahn et al. 1999, Roy and Kodkani 2000, Chen and Jan 2000, Domazet et al. 2000]. Product development techniques such as design for manufacturing (DFM) can be enhanced by web technology [Park and Baik 1999, Jiang

and Fukuda 1999]. Product data exchange is also a field for applications of Internet technologies [Zhang et al. 2000].

Concurrent product development is a subject involving a wide range of concepts, methods, and technologies. The advantages of concurrent design have been recognized, but implementations of concurrent design systems need to be further explored. Feature-based product modeling method promises to be effective in product development systems. Multi-agent systems can reasonably be regarded as powerful tools for developing future complex product development systems.

However, the difficulties in developing "intelligence" have limited the applications of multi-agent systems in actual product development system implementations. Most of the Internet and web-based systems for product development focus on browsing product geometric descriptions from remote product databases and exchanging information among designers. The real advantage of the Internet technology is the properties of an open network and real time communications. Therefore Internet-based product development systems can go one step further in integrating distributed, especially geographically distributed, product development activities in all aspects of product life-cycle.

2.2 Related Techniques

In this research, feature-based modeling was used as the approach of modeling product and related processes. The feature-based distributed modeling system was implemented using an object-oriented programming technique. VisualWorks version of Smalltalk, an object-oriented programming language, was used in this research. Distributed modeling was the basic objective of this research. To improve the quality of product development, engineering optimization methods including Genetic Programming (GP) and Particle Swarm Optimization (PSO) were used in identifying the optimal product realization process. These techniques are briefly introduced in this section.

2.2.1 Feature-Based Modeling

Conventional solid modeling is efficient for defining the geometry of a product. However there are two major deficiencies, according to Shah and Rogers [1988]:

- Product definition is incomplete: Product tolerance, surface finish, surface treatment, and other descriptions apart from geometry, cannot be represented and stored.
- Product definition is at a lower level: The product data are basically for displaying the image of the product. Higher level properties such as functions cannot be defined.

In order to use the product model to develop applications for manufacturability evaluation and process planning, feature-based modeling was introduced. Based on Shah and Rogers [1988], a feature is a set of information related to a description of a part or a product. The description may be used for design purposes, marketing requirements, manufacturing process development, assembly, inspection, and even administrative purposes. By using feature-based modeling, the product models can be built using features stored in the libraries.

There are different classifications of features. Based on the information sets related to the product engineering, there are form features, material features, precision features, and technological features, etc. [Shah and Rogers 1988]. Features can also be classified into design features, manufacturing features, assembly features, etc. according to the life-cycle functions of the features [Xue and Dong 1993].

Feature-based modeling provides a means for building product databases at multiple abstraction levels. According to Chung et al. [1990], feature-based modeling is efficient in the following ways:

- Human intent can be expressed easily by manipulating both high and low level features directly.

- Feature databases allow the reasoning system to perform product development tasks such as manufacturability evaluation, function analysis, and design optimization.
- Feature databases can contain knowledge to facilitate more applications such as CNC programming.

Some applications of the feature-based approach in product development were introduced in Section 2.1.1. A more detailed explanation of feature-based modeling concept will be given in Section 2.3.

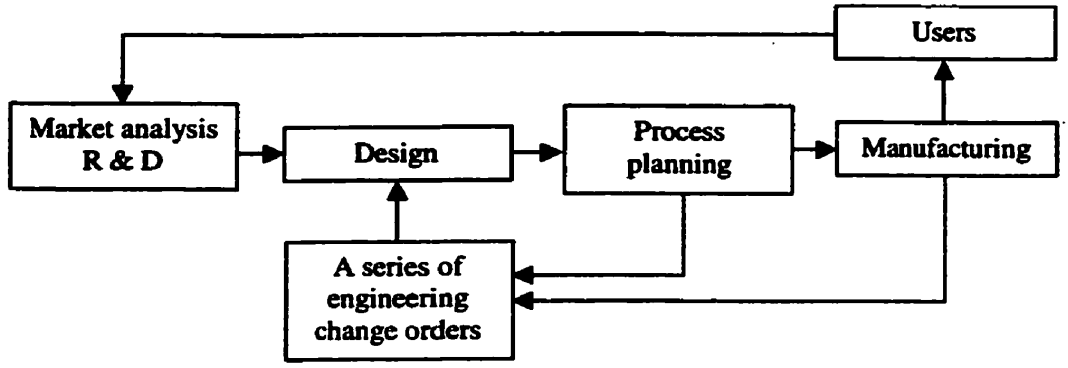
2.2.2 Concurrent Engineering

Conventionally, product development follows a sequential development cycle, as illustrated in Figure 2.1 (a). The cycle begins with a need based on market analysis or research and development results. Then the product is developed step by step through design, process planning, manufacturing, assembly, and shipping. In this approach, design concerns are mainly focused on the functionality and performance of the product. Very few requirements of down-stream life-cycle phases are considered, since there is no dialogue established between design and down-stream processes. However most down-stream performance of a product is determined at the design stage. For example about 70-80% of manufacturing productivity can be determined at the design phase [Suh 1990].

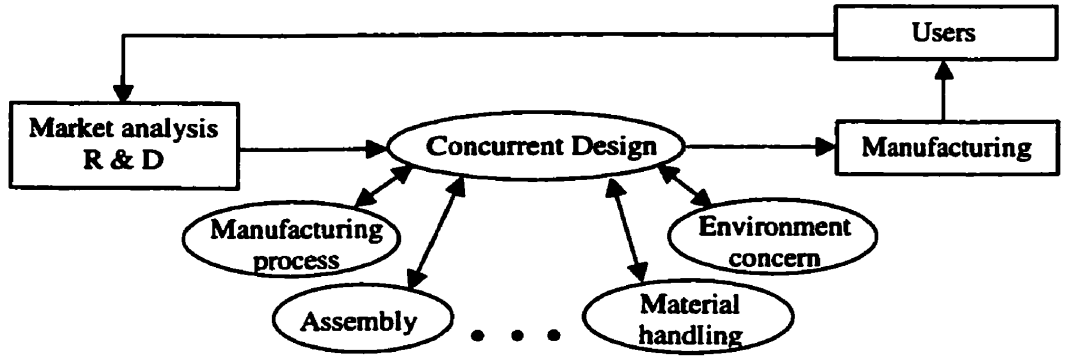
In a global competitive economy environment, a product with high quality, low cost, and less environmental impact can be achieved with a new product development philosophy called Concurrent Engineering (CE) [Kusiak 1993, Prasad 1996].

In developing product using a concurrent engineering or concurrent design approach, all related processes such as marketing, design, process planning, manufacturing, assembly, and recycling are considered concurrently, as illustrated in Figure 2.1 (b). The design activities in concurrent engineering have been widely extended [Hyeon et al. 1993] to the related processes. In other words, down-stream requirements should be considered as early as possible, along with the structural and functional requirements of

the products. To implement this approach, organization-wide, even global-wide information integration is required.



(a) Sequential Product Development



(b) Concurrent Product Development

Figure 2.1 Sequential and Concurrent Product Development [Hyeon et al. 1993]

Currently there are two basic approaches to implementing concurrent engineering practice: the team-based collaboration approach and the computer-based development approach. The former approach emphasizes information flows among designers and individuals from all related areas. The multifunctional team is critical for effective implementation of the concurrent development of products. The information flowing among team members can be assisted by computer systems. The computer-based approach enables design selection, justification, and optimization with respect to all aspects of a product's life-cycle. Therefore this approach emphasizes direct cooperation and coordination among design and all related down-stream processes that are

represented as models to be handled by computers. Knowledge-based approaches are often employed in a computer-based concurrent engineering environment [Court 1998]. With the rapid progress of Internet techniques, more attention has been paid to developing Internet-based distributed systems for concurrent engineering [Roy et al. 1997].

2.2.3 Object-Oriented Programming and Smalltalk

Programming languages are traditionally composed of two parts – the data, and operations on the data. For a procedural programming language like C, functions and data structures are the basic elements. Object-oriented programming groups operations and data into modular units called objects, and lets you combine objects into structured networks to form a complete program. In an object-oriented programming language, the objects and object interactions are the basic elements.

The four basic characteristics of object-oriented programming are Abstraction, Encapsulation, Polymorphism, and Inheritance. Abstraction refers to the essential characteristic of an object that distinguishes it from other objects. Encapsulation keeps the implementation of an object out of its interface. In other words, details of implementation are hidden from other parts of the programs. Encapsulation not only protects an implementation from unintended actions but also increases the modularity of the program. Polymorphism refers to the ability of different objects to respond, each in its own way, to identical messages. The main benefit of polymorphism is that it simplifies programming interfaces. Instead of creating a new name for each new function to be added to a program, the same names can be used by many objects. Inheritance is the feature that an object can be defined based on an existing object and the characteristics of the existing object are passed on to the new object automatically. Inheritance clarifies the logic relations of objects. This property also brings benefits such as reusing code and delivering generic functionality.

Object-oriented programming techniques are implemented using the concepts of class and instance. Classes are generic abstractions of physical objects with similar

characteristics. An instance is an object with specific attributes. An instance is created using a class as the template. Inheritance is implemented by class-subclass relation definitions.

Available object-oriented programming languages include Smalltalk, C++, Java, etc. Smalltalk, as a pure object-oriented programming language, was developed in the late 1970s [Goldberg and Robson 1983, Hopkins and Horan 1995]. The Smalltalk system has two aspects: the programming language and the programming environment. Smalltalk was one of the first systems to use graphical interfaces to help the user navigate the development system. In the Smalltalk environment, everything is an object. All the functions of the system including file handling, compiling, debugging, window managing, etc. are defined using classes and their instances. In the latest versions of Smalltalk, a large number of classes have been provided for developing applications. New classes can be defined as subclasses of the existing classes.

There are several dialects of Smalltalk such as VisualWorks, SmalltalkAgents, VisualAge, etc. VisualWorks has been relatively widely used. The syntax of Smalltalk is descriptive and the rules of the syntax are simple. This feature helps shorten the time of developing Smalltalk applications. It has been proven that Smalltalk is effective for developing research-oriented application prototype systems [Xue et al. 1992].

2.2.4 Distributed Systems and the Internet

A distributed system is a collection of independent computers associated through both hardwares and softwares. Generally, a distributed system means a close coordination among components at different sites [Wu 1999]. Distributed systems are usually composed of distributed hardwares, distributed data, and distributed controls. A distributed system includes nodes that perform some aspects of computations. A node may be a personal computer, or a mainframe computer. The nodes of a distributed system are usually geographically distributed. The node you currently use is regarded as the local node and all others are remote nodes. The power of a distributed system derives from the cooperation of the individual nodes that perform different functions.

The nodes in distributed systems are connected by computer networks. The advance of the Internet has been a force driving distributed systems forward in recent years [McCarty and Cassady-Dorion 1999].

The Internet is considered a global computer network that connects groups of sub-networks. These networks contain many different types of computers. A protocol must be used to ensure that the different types of computers can work together. A protocol is a set of rules that specify how computers cooperate in exchanging messages [Hahn and Stout 1994]. TCP/IP (Transmission Control Protocol/Internet Protocol) is the most popular networking standard. TCP/IP is used to organize computers and other communication devices into a network. IP transmits the data from place to place, while TCP formats the data and manages the flow.

In the network, each computer must have an address in order to be located by other computers. There are two formats: standard address and IP address. For example: the standard address m70.enme.ucalgary.ca is equivalent to the IP address 136.159.105.70. The IP address is the real address used in identifying the computer. The standard address can be translated into the IP address by the Domain Name Service (DNS).

In order for TCP to locate a specific process of an application in a computer, port numbers are used to specify this process. Port numbers are 16-bit numbers such as 3456. Some port numbers, called well-known port numbers, are reserved for standard applications such as mailing services. The remaining ports are dynamically allocated ports for implementing sockets. The combination of an IP address and a port number can identify the required program.

Standard Internet technologies, including WWW (World Wide Web), E-mail, and VRML (Virtual Reality Modeling Language), have been studied for the purpose of assisting product development. However the real power of the Internet is the feature of an open network and real time global connections. Internet makes it possible for globally scattered computers to work together dynamically.

2.2.5 Global Optimization

Optimization is an approach used to identify the optimal solution for a problem based on predetermined objectives. An optimization problem is made up of three basic ingredients:

- An objective function which we want to minimize or maximize. For example, in a design problem, we might want to maximize the product life span or minimize the cost.
- A set of variables that affect the value of the objective function. In mechanical design problems, the speed of a rotating part and the distance between the centers of two shafts are typical variables.
- A set of constraints that allow the variables to take on certain values but exclude others. For engineering problems, the length of a part cannot be negative, so this variable should be constrained to be positive (or between two positive numbers).

The optimization problem is then to find values of the variables that minimize or maximize the objective function, while satisfying the constraints. Since many engineering problems have local optimums, traditional approaches such as the hill-climbing method may miss the global optimum [Arora et al. 1995]. Global optimization is the task of finding the absolutely best set of conditions within the constraints to achieve the objective. There are two basic categories of global optimization approaches: deterministic and stochastic [Pardalos et al. 1999].

Deterministic approaches exploit analytical properties of the given problem to generate a deterministic sequence of conditions that converge to the global optimal solution. Stochastic approaches minimize a function over a random set of variable values. These approaches can be used for problems when no clearly known structure can be exploited. In the past decade, some stochastic approaches such as Simulated Annealing, Genetic Algorithms, and Particle Swarm Optimization have been studied and effectively applied to a wide range of industry applications [Arora et al. 1995, Shi et al. 1997, Pardalos et al. 1999].

2.3 A Feature-Based Database and Knowledge Base Representation Scheme

The research on integrating databases and knowledge bases was started by Xue at the University of Tokyo during the development of the Integrated Data Description Language (IDDL) [Xue et al. 1992]. Xue had used IDDL to implement the previous version of the feature-based concurrent design system at the University of Victoria [Xue and Dong 1993, Xue and Dong 1994]. A complete new feature modeling environment was developed by Xue et al at the University of Calgary [Xue et al. 1999]. The feature-based database and knowledge base representation scheme introduced by Xue at the University of Calgary was employed in the research discussed in this thesis.

2.3.1 Database Representation

2.3.1.1 Class Features and Instance Features

Product life-cycle aspect models are built using primitives called features. Features are described at two different levels, the class level and the instance level, corresponding to generic product libraries and specific product data respectively, as shown in Figure 2.2. Instance features are generated using the class features as their templates. This mechanism was implemented using an object-oriented programming approach.

A class feature is defined by element-features, attributes, qualitative relations among features, and quantitative relations among attributes. Element-features are described by variables and their feature types, representing the features used to compose the feature being defined. For instance, the ThreadHole class feature shown in Figure 2.2 consists of two element features, a hole and an internal thread, represented by two variables ?H and ?IT. The class feature itself is described by a built-in variable ?self in the class feature definition. Attributes in class features are defined by attribute names and default attribute values. For instance, diameter, d, and length, l, are two attributes of the class feature Hole shown in Figure 2.2. Qualitative relations among features are represented by predicates. A predicate takes the form of $(p, x_1, x_2, ..., x_n)$, where p is the predicate relation and $x_1, x_2, ..., x_n$ are terms of this predicate represented by symbols (e.g., h1), strings (e.g., "Hello"), integers (e.g., 5), floats (e.g., 2.5), variables (e.g., ?H), and

attributes (e.g., d[h1]). The predicate (process, ?H, ?IT) in class feature ThreadHole is a qualitative relation among features. Quantitative relations among attributes are described by functions. Each function uses a number of input attribute values to calculate an output attribute value. For instance, the function, l[?H] := l[?IT] + 5, in class feature ThreadHole is a quantitative relation of two attributes.

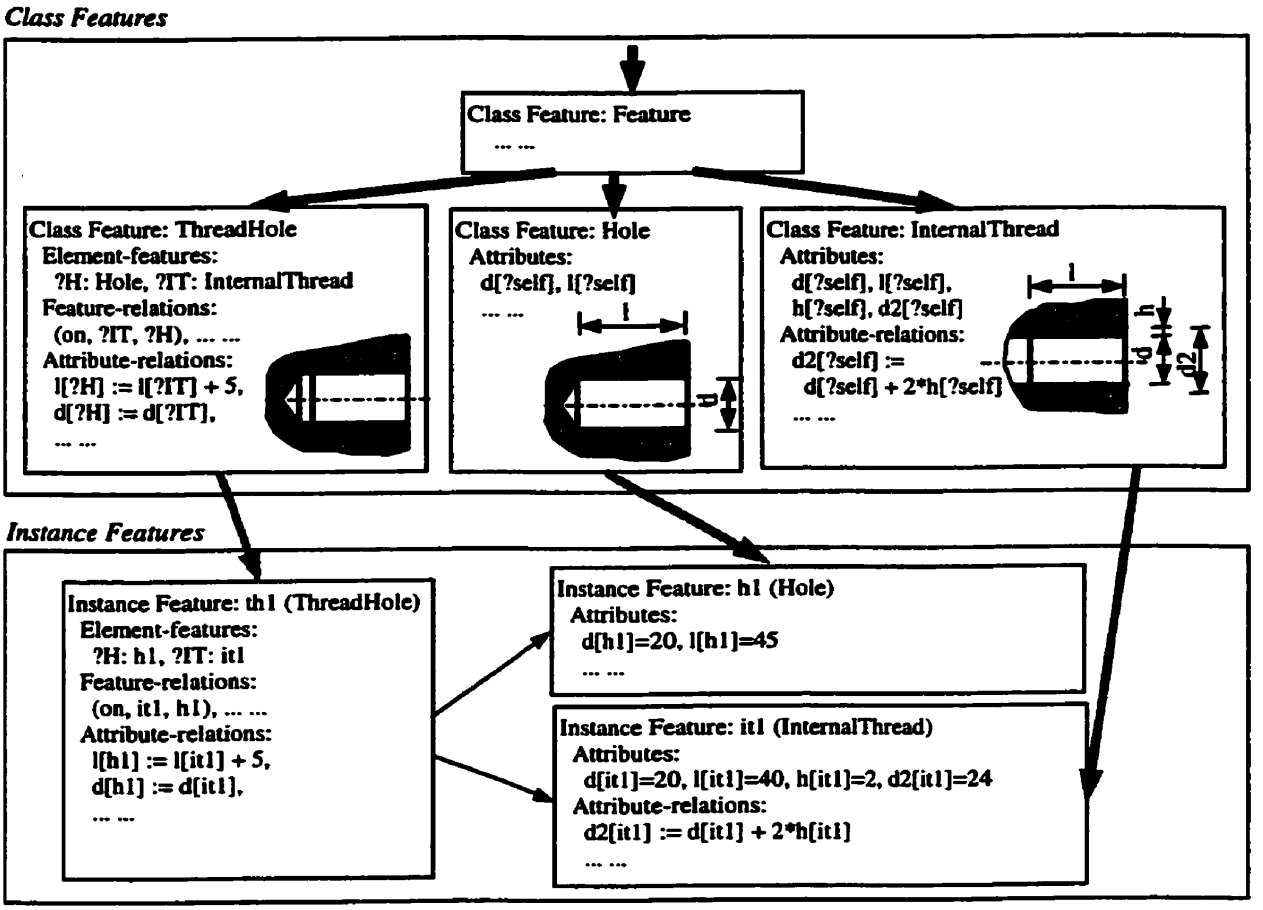


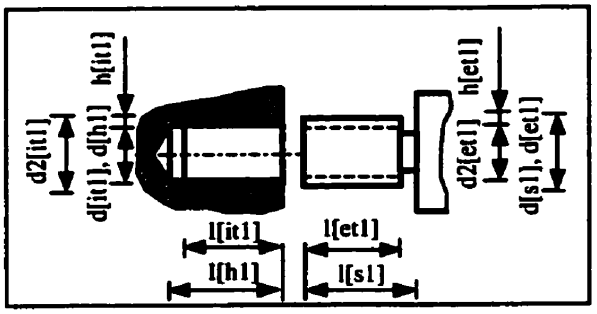
Figure 2.2 Class Features and Instance Features

Class features are organized in a hierarchical data structure. A class is defined as a sub-class of an existing super-class. All descriptions of a super-class are inherited by its sub-classes automatically. When an instance feature is generated using a class feature as the template, all the descriptions in that class feature and its super-class features should be added to the database automatically. In an instance feature, all the element-feature variables are instantiated by actual instance features with the required feature types. In

Figure 2.2, three instance features, representing a thread-hole of a product, are generated from three class features.

2.3.1.2 Maintenance of Data Dependency Relations

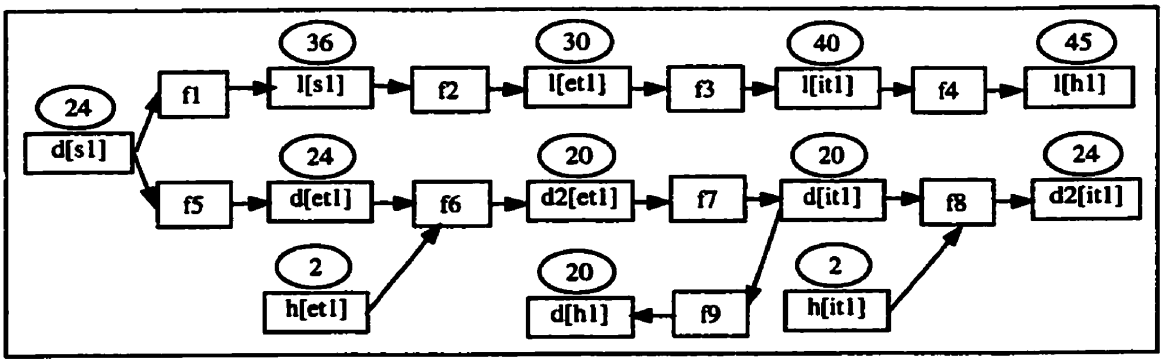
The quantitative relations among attributes in the generated instance features form a network called *attribute relation network*. An example of attribute relation network is shown in Figure 2.3. An attribute relation network is composed of two types of nodes: attribute nodes and function nodes. Each attribute node is associated with an attribute value. Each function node is linked with one or several input attribute nodes and one output attribute node. When an attribute value is changed, the functions that use this attribute as the input node are then activated to update this change to the output attribute nodes. This attribute propagation process is carried out continuously until no attribute value change is required. Since the attribute relation network can be used for keeping the consistency of the database, the mechanism to update attribute changes using the attribute



(a) A Thread Joint

f1: l[s1] = 1.5 * d[s1]
f2: l[et1] = l[s1] – 6
f3: l[it1] = l[et1] + 10
f4: l[h1] = l[it1] + 5
f5: d[et1] = d[s1]
f6: d2[et1] = d[et1] – 2 * h[et1]
f7: d[it1] = d2[et1]
f8: d2[it1] = d[it1] + 2*h[it1]
f9: d[h1] = d[it1]

(b) Attribute Relations



(c) An Attribute Relation Network

Figure 2.3 Maintenance of Data Dependency Relations

relation network is called the *data dependency relation maintenance mechanism.*

Propagation of attribute value changes using the attribute relation network is formulated into the following steps.

Step 1: Create a list called ATTRIBUTE-CHANGE and add all the changed attributes to this list.

Step 2: If the ATTRIBUTE-CHANGE list is empty, attribute change propagation should be stopped.

Step 3: Remove one attribute from the ATTRIBUTE-CHANGE list. Identify the functions that use this attribute as an input attribute, and calculate the output attribute values using these functions. If the value of an output attribute is changed and this attribute is not on the ATTRIBUTE-CHANGE list, add this attribute to the list. Go to Step 2.

This attribute relation network is effective for modeling the relations defined in the databases for product development.

2.3.2 Knowledge Base Representation

The knowledge base is represented by rules. Since a product development process involves a large number of rules, these rules are organized in separated rule-bases. During the product development process, only partial rule-bases are considered to improve the inference efficiency. All rule-bases are preserved in the rule-base library.

2.3.2.1 Rule-Bases

A rule-base is defined by a rule-base name and a collection of rules, as shown in Figure 2.4. Each rule description is composed of a rule name and the rule itself. A rule takes the form of IF-THEN data structure, representing a piece of cause-result knowledge. Both the IF part and the THEN part of a rule are represented by a number of patterns linked with logical-and (&). A pattern is described by a predicate using the form of $(p, x_1, x_2, ..., x_n)$, where p is the relation and $x_1, x_2, ..., x_n$ are terms. Terms are represented by symbols, numbers, variables, and attributes, as introduced in Section

2.3.1. The condition part and the result part of a rule are used for matching, creating, deleting, and modifying the data in the product databases, including features, attributes, qualitative relations among features (facts), and quantitative relations among attributes (functions). In the rule-base shown in Figure 2.4, the built-in predicates, featureType, assertFeature, and =, are used for matching the class types of instance features, creating instance features, and adding functions, respectively.

```
Rule-base: FeatureManufacturingProcess

Rule: DrillingProcess
IF (featureType, ?X, Hole)
THEN (assertFeature, ?Y, Drilling) & (=, d[?Y], d[?X]) & (=, l[?Y], l[?X]).

Rule: ThreadingProcess
IF (featureType, ?X, InternalThread)
THEN (assertFeature, ?Y, InternalThreading) & (=, d[?Y], d[?X]) & (=, l[?Y], l[?X]).

... ...
```

Figure 2.4 A Rule-Base

2.3.2.2 Reasoning with Rule-Bases

In a feature-based database and knowledge base modeling system, the databases are described by features and the knowledge bases are described by rule-bases. Since usually a large number of features and rule-bases are used for modeling the development of a product, a mechanism to select only partial database and knowledge base has to be developed to improve the computation efficiency. In the feature-based database modeling system, since an instance feature is composed of element features, attributes, qualitative relations among features, and quantitative relations among attributes, an instance feature can be selected as such a partial database considered in knowledge-based inference. The partial knowledge base considered in inference is the rule-bases selected from the rule-base library for the selected instance feature. Therefore, each instance feature is associated with a number of selected rule-bases. This idea is illustrated in Figure 2.5.

The product modeling using knowledge-based reasoning approach starts with selecting an instance feature as the active instance feature. For this active instance, a number of rule-bases are selected from the rule-base library. All the rules in the selected

rule-bases are registered in the active instance feature. The inference is carried out first by matching the condition parts of all the registered rules with the active instance feature database. If multiple rules are matched, the best rule is selected and the result part of this rule is executed. In this research, the first matched rule is considered the best rule to be fired. This matching/execution process is carried out continuously until no rule can be matched.

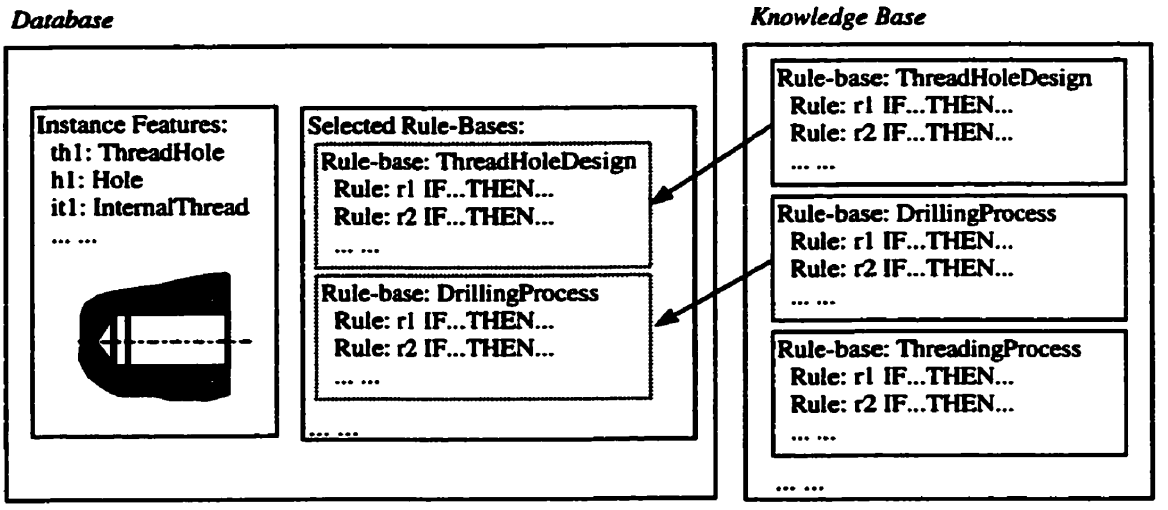


Figure 2.5 Selection of Partial Knowledge Base and Database for Reasoning

CHAPTER 3

DISTRIBUTED PRODUCT DATABASE AND KNOWLEDGE BASE MODELING

This chapter introduces the distributed feature-based product database and knowledge base modeling system. Following the introduction of Section 3.1, the architecture of the distributed database and knowledge base modeling system is presented in Section 3.2. Section 3.3 provides a detailed discussion on distributed database modeling. First, the concepts of virtual features including virtual class features and virtual instance features are introduced. Then, the methods for associating the distributed databases by defining relations among true features and virtual features are described. For automatic data dependency relation maintenance, an algorithm for propagating data changes to related data preserved in accessible remote nodes is given. Section 3.4 discusses issues in distributed knowledge base modeling for product development. These issues include modeling of virtual rule-bases and the distributed knowledge-based inference.

3.1 Introduction

Conventional product development follows sequential procedures from marketing, design, manufacturing, and assembly to shipping and service. The life-cycle performance of the product designed using this approach is not optimal because of insufficient information exchanges among these life-cycle development activities during the design process. Concurrent design approach considers relevant product development processes concurrently. Since there are mutual information flows between design and related down-stream development processes, the product design using concurrent engineering methodology improves life-cycle performance of the product. The different product development activities are usually geographically distributed. With the increasingly competitive global market, incorporation of the geographically separated product development resources is required to improve the overall performance of the products.

The Internet technique provides a unique tool for integrating distributed computer systems. It allows people and computer systems to communicate dynamically in a global computing environment. The low cost of connecting to an Internet service also makes it advantageous to use the Internet as the medium for connecting product development activities. In this research, the distributed product development activities are associated using the Internet.

3.2 Distributed Product Database and Knowledge Base Modeling Architecture

To develop the distributed product database and knowledge base modeling system, two issues have to be addressed: modeling of product development activities, and the association of these activities.

3.2.1 Product Development Life-Cycle Activity Modeling

The product modeling technique is important to the effectiveness of computer-based product development systems. To incorporate concurrent design methodology, modeling of product development activities at different development phases such as design, manufacturing, recycling, etc. is required. These activity descriptions are used for modeling both geometric and non-geometric properties of products. In this research, feature-based modeling technique is employed [Shah and Rogers 1988, Xue et al. 1999, Yadav 1999]. The feature-based modeling approach was introduced in Chapter 2.

In this research, modeling of activities of the product development life-cycle is emphasized. Typical activities include marketing, design, manufacturing, service, and recycling [Singh 1995]. The features employed for modeling the product development processes are described at two levels: class level and instance level. Class features represent generic product development libraries. Instance features are actual databases of specific product development activities. Instance features are created using class features as their templates. Figure 3.1 illustrates the class features and instance features for modeling shaft-manufacturing process.

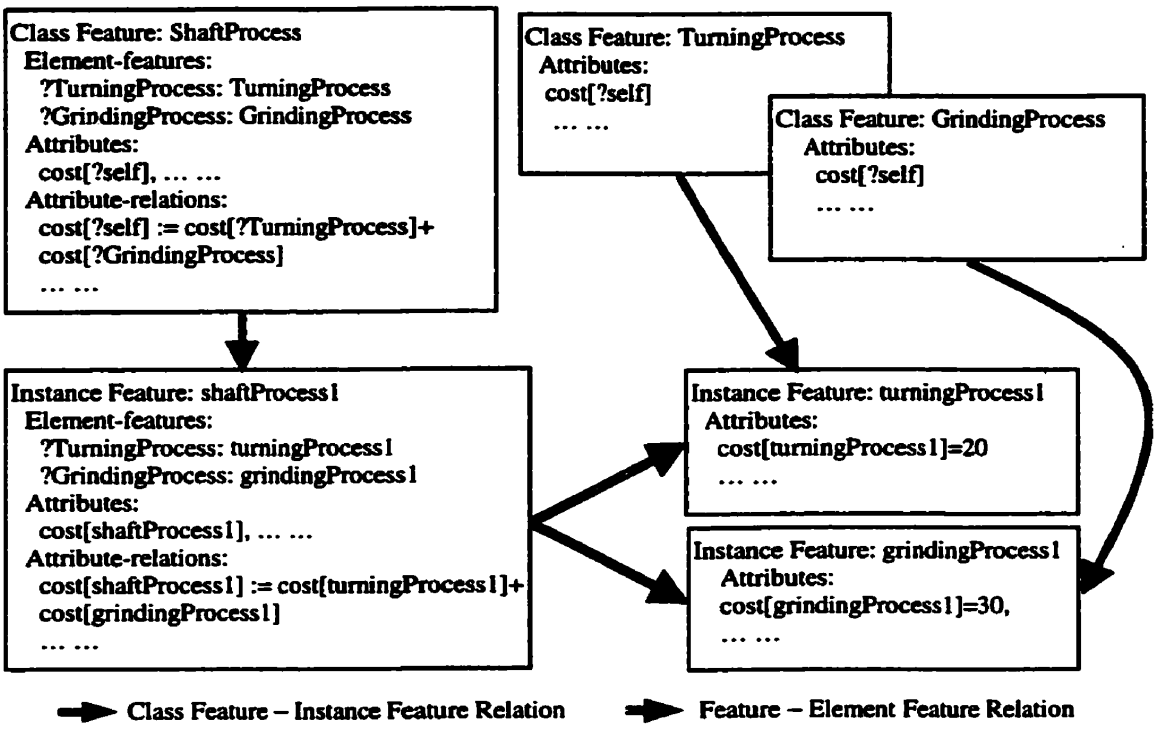


Figure 3.1 Modeling of a Shaft Manufacturing Process

Since the product development life-cycle activities are described using features preserved at different locations, modeling of the relations among these activities must be incorporated into the developed system. Details about the relation modeling are introduced in Section 3.3.

3.2.2 The Client-Server Communication Architecture

The integration of distributed product development models is accomplished through the Internet. The integrated system is an Internet-based computer network system. One common architecture for a computer network has at least three basic components [McCarty and Cassady-Dorion 1999]: a client, a server, and the network itself, as shown in Figure 3.2.

Usually there are many clients in a computer network. The network associates the clients with the server. The clients are usually operated by users to request information from the server. A server holds resources needed to satisfy the client requests. Clients'

requests flow through the network to the server, and the server's responses flow across the network to the clients. In this research, the Internet is the medium for connecting the clients and the servers.

Figure 3.2 Three Basic Components of a Computer Network

In product development with concurrent design methodology, information flows in multiple ways among the different development processes at different locations. This requires the computer, used for modeling product development activities in certain phases, to be both a client and a server. In such cases, client and server become roles in a logical sense rather than physical devices. Therefore, not only can a server have many clients; a client can also connect to many servers.

The computers connected to the Internet can be called Internet nodes. In this research project, communication among the Internet nodes is implemented using socket-based client-server architecture [Hahn and Stout 1994]. Sockets are computer programs that let you send and receive messages among networked computers. As a data exchange tool, sockets are simple to use and operate efficiently. The concurrent design methodology requires that information flow in and out of the Internet nodes. Therefore, each node can be a client or a server depending on the direction of information flow during product modeling processes. So all the Internet nodes run both client side and server side socket programs, in terms of sending and receiving messages.

In the example shown in Figure 3.3, all computers at different locations are connected to the Internet. Each node can run both a Smalltalk server socket and a client socket. A node, (e.g., the node B), can be both a server and a client. While the server socket of node B is running, the other nodes A, C, and D are then the clients of node B, so they can request information from node B. The node B can also request information

from other nodes. On such occasions, all other nodes A, C, and D are servers of node B, and node B is a client.

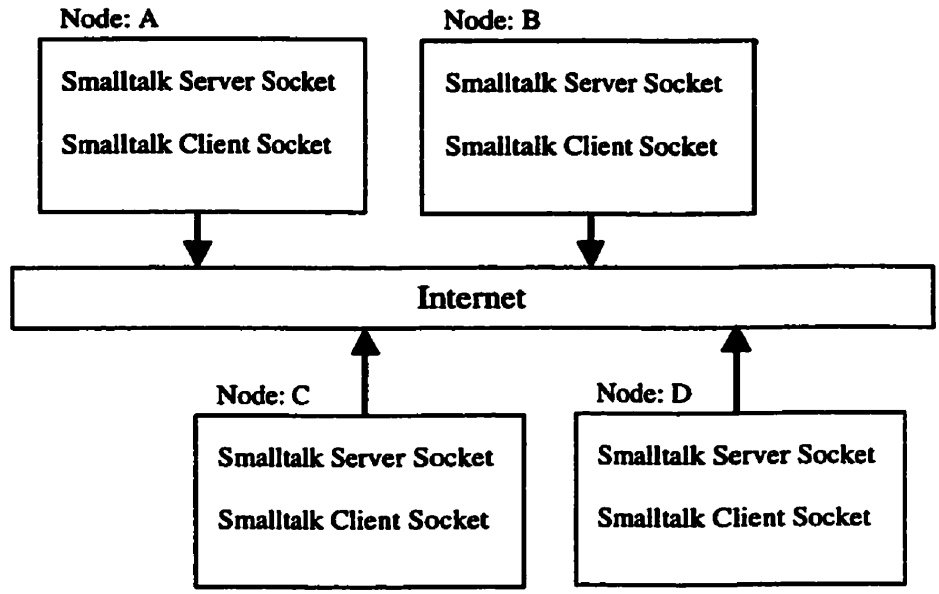


Figure 3.3 Logical Clients and Servers

3.2.3 Architecture of the Distributed Database and Knowledge Base Modeling System

The architecture of the distributed database and knowledge base modeling system is shown in Figure 3.4. In this architecture, different databases and knowledge bases used during different product development phases, including marketing, design, manufacturing, etc., are modeled at different locations represented as nodes, such as Marketing1, Marketing2, Design1, and so on. Since the databases and knowledge bases are linked by the Internet, these nodes are also called *Internet nodes*. During the product development process, the database and knowledge base accessibility relations among these nodes are first defined for collaboration in concurrent design. When node A is defined to be able to access node B, all the data and knowledge in node B can be used in node A automatically. Since the distributed database and knowledge base modeling method associates different product development activities at different locations into an integrated environment, this approach can evaluate down-stream product development aspects during the early design stage, thus improving product development efficiency and

quality.

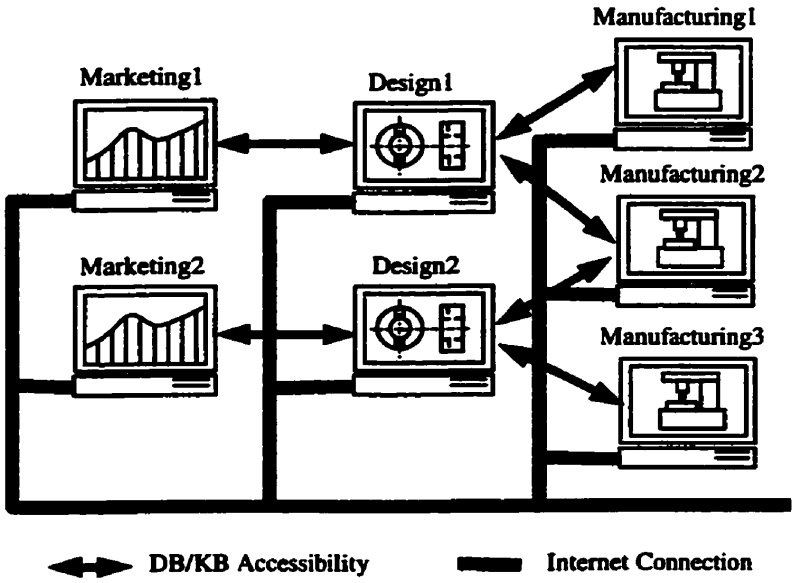


Figure 3.4 Architecture of the Distributed Database and Knowledge Base Modeling System

In the example shown in Figure 3.4, the Marketing1 node and the Design1 node are defined to be mutually accessible. When certain requirements for products are identified from customers at the Markerting1 node, these requirements are then used as the guidelines for creating and improving product designs at the Design1 node. The designs are then evaluated at the Marketing1 node to see whether the customer requirements have been satisfied. If a design created at Design1 node doesn't satisfy the customer requirements at Marketing1 node, the accessibility relation between Marketing1 node and Design1 node is removed. A new accessibility relation between Marketing1 node and Design2 node can then be established to generate another candidate at Design2 node to satisfy the customer requirements at Marketing1 node. The design node Design1 is linked with two manufacturing nodes Manufacturing1 and Manufacturing2 for evaluating the manufacturability of the design and using the evaluation measures to improve the design.

Each node in the Internet is specified by its address and port number. A node address can be described either by an Internet Protocol (IP) address, such as 136.159.105.72, or by a standard address, such as m72.enme.ucalgary.ca. A port number is a 16-bit digit,

such as 9876. Examples of Internet node definitions are shown in Figure 3.5.

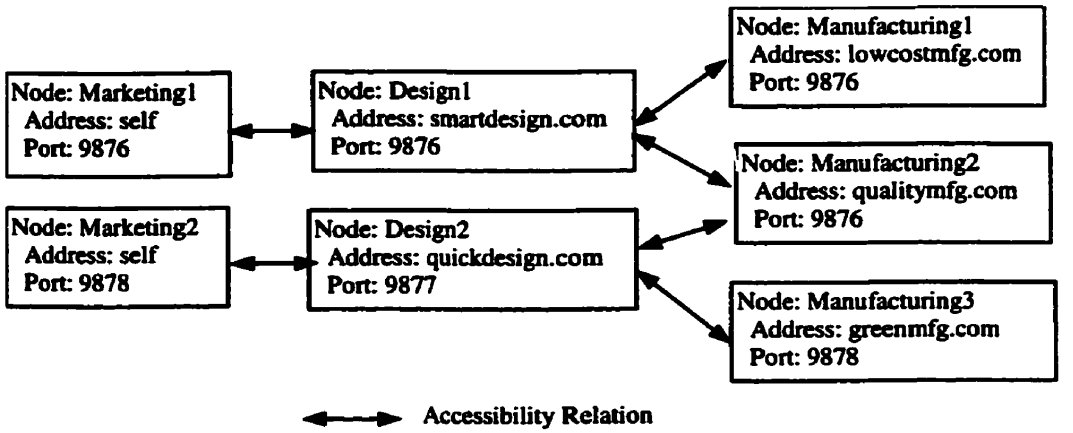


Figure 3.5 Definitions of Internet Nodes

When a node A is defined to be able to access node B, all the data and knowledge preserved in node B can be used by node A automatically. The data and knowledge in node B are considered as virtual data and knowledge in node A. This idea is illustrated in Figure 3.6. The database and knowledge base accessibility relation between two nodes is implemented using client-server communication architecture. In this architecture, a node to access other nodes is a client that sends messages to the accessible nodes for obtaining the information of available data and knowledge preserved in these accessible nodes, and a node to be accessed by other nodes is a server that responses messages from the client nodes for providing available data and knowledge to these client nodes.

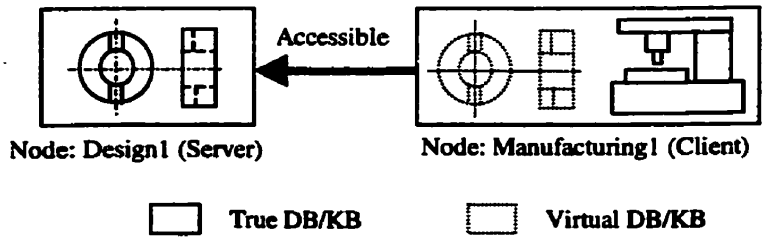


Figure 3.6 An Accessibility Relation between Two Nodes

Distributed database and knowledge base modeling architecture is employed to link the feature-based product development life-cycle activities into an integrated environment to improve the product development efficiency. Details regarding the

feature-based distributed database modeling and knowledge base modeling will be given in Sections 3.3 and 3.4.

3.2.4 Node Definitions and Node Connections

The Internet serves as the tool for connecting the Internet nodes involved in this distributed product database and knowledge base modeling system. For socket-based client–server communication, both Internet addresses (Standard addresses or IP addresses) and port numbers are required to identify the target programs running on different computers. Before communication can be conducted among Internet nodes, the corresponding addresses and port numbers of these Internet nodes have to be defined first. In this distributed database and knowledge base modeling system, two browsers, the Internet Node Definition Browser and the Node Connection Browser, are used to handle node definitions and connections.

3.2.4.1 Definition of Internet Nodes

The Internet nodes are defined by their Internet addresses and port numbers. The Internet Node Definition Browser is used for defining the Internet nodes involved in the product development processes. Hardcopy and views of this browser are shown in Figure 3.7 and 3.8. There are three views in this browser: A, B, and C. A is the category list view, B is the node name list view, and C is the text view. The categories of the nodes are listed in the category list view. The nodes are grouped in the categories. When one of the categories is selected, the nodes in that category are shown in the node name list view. When a node name in the node name list view is selected, the node descriptions, including node name, address, and port number are shown in the text view. The text view is a text editor for editing the node information, including the node name, address, and port number.

The menus of these views are also shown in the Figure 3.8. The commands of these menus are mainly used for editing, adding, and deleting node definitions.

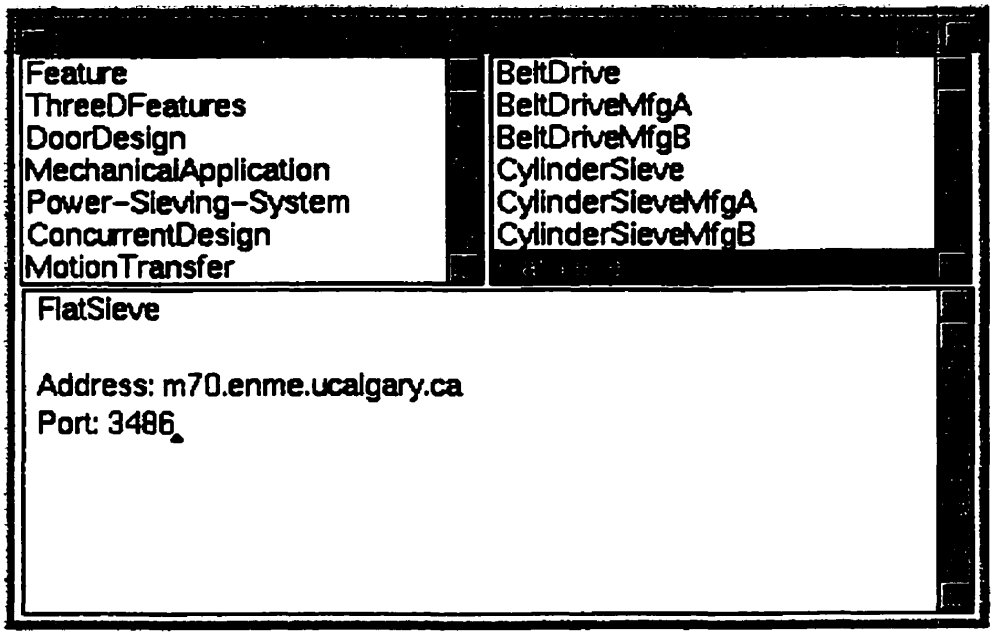


Figure 3.7 A Snapshot of the Internet Node Definition Browser

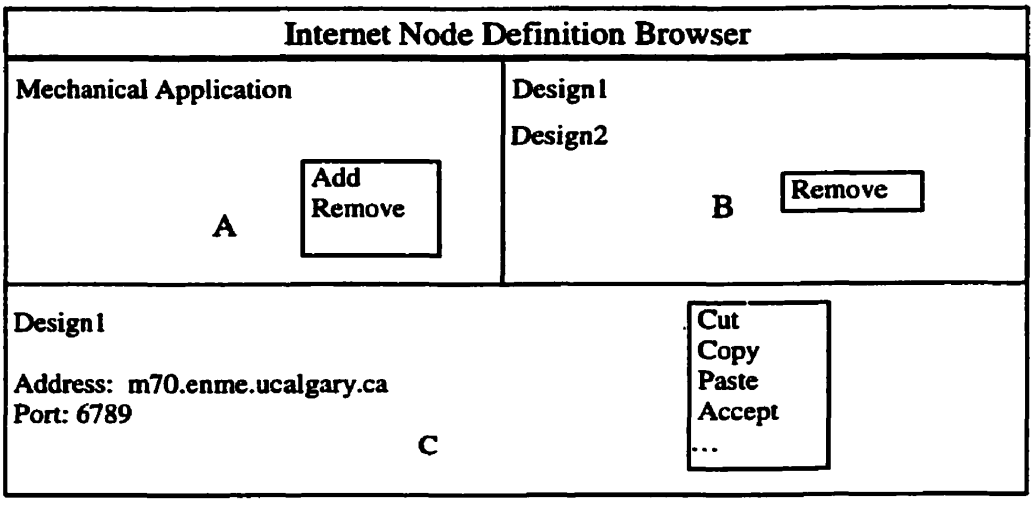


Figure 3.8 Configuration of the Internet Node Definition Browser

The typical procedure for defining an Internet node is:

a. Create a new category by selecting Add command of the category list view menu, or select an existing category.

b. Under this selected category, edit the descriptions of a node, including node name, address, and port number, following the required format in the **text view**. Select **Accept** menu item to save the edited descriptions. The node name will appear in the **node name list view**.

3.2.4.2 Connection of Internet Nodes

Connecting and disconnecting a node are accomplished through the **Node Connection Browser**. This browser, as shown in Figure 3.9 and 3.10, allows the user to connect the local node to remote nodes by highlighting a node name and clicking **Connect** in the menu. If the connection is successful, the letter "C", representing *Connected*, appears after the node name in the node name list view. Before a node can be

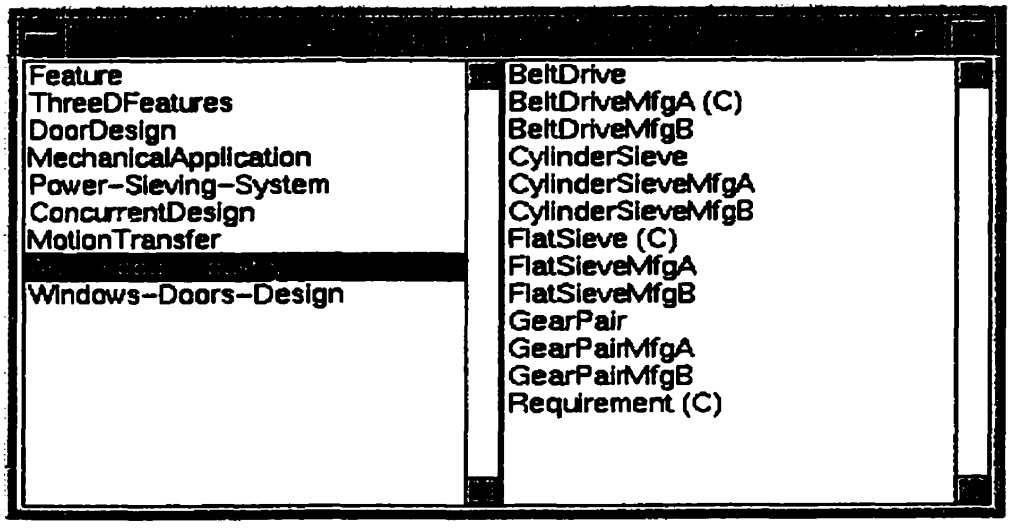


Figure 3.9 A Snapshot of the Node Connection Browser

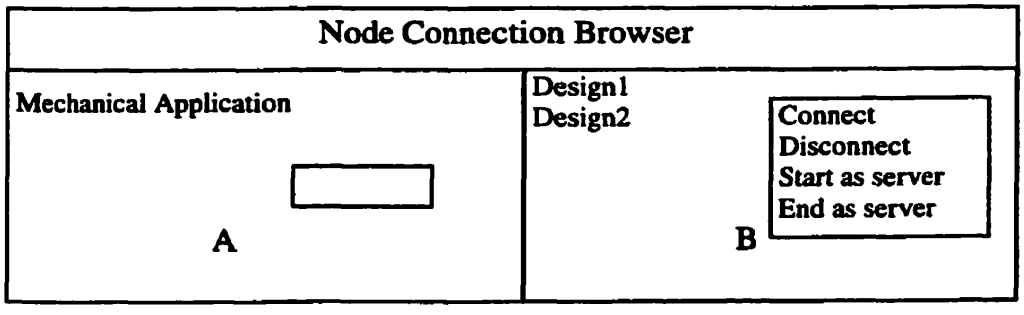


Figure 3.10 Configuration of the Node Connection Browser

connected to a server node, the server program in that server node must be executed. A server process can be started by clicking on **Start as server** in the menu. There are two views in this browser: A – the **category list view**, and B – the **node name list view**. These views are identical to the ones in the **Internet Node Definition Browser**. When a category in the category list view is selected, node names in that category are listed in the node name list view. Those nodes that have been already defined in the system are ready to be connected.

3.3 Distributed Feature-Based Database Modeling

In the distributed feature-based database modeling approach, the class features and instance features at accessible remote nodes are considered as virtual class features and virtual instance features at the local node. Virtual class features can be used for generating instance features at the local node. Virtual instance features are considered as part of the database at the local node and can be accessed from the local node. By defining the dependency relations among the data distributed at different locations, the consistency of the product development databases can then be maintained using these relations.

3.3.1 Virtual Features

In this research, databases for modeling product development activities are described by features. These features are modeled in different Internet nodes at different locations. When node A is defined to be able to access node B, the features preserved in node B are considered as virtual features in node A. Virtual features are of two types: virtual class features and virtual instance features.

3.3.1.1 Virtual Class Features

Virtual class features are class features preserved in accessible remote nodes. They represent generic libraries of different product development life-cycle aspect databases. During the product development processes, the instance features, representing the actual product databases, are generated, using corresponding class features as their templates. If

the required class features cannot be found at the local node, virtual class features at accessible remote nodes can be used for generating the true instance feature at the local node. This characteristic can improve the efficiency of product development by sharing library resources among all accessible Internet nodes.

A virtual class feature is defined by the node name and the class feature name in the form of:

<node name>%<class feature name>

The concept of virtual class features is illustrated in Figure 3.11. The class feature Shaft in node A is described as A%Shaft in node B. The three class features in node A are considered as virtual class features in node B.

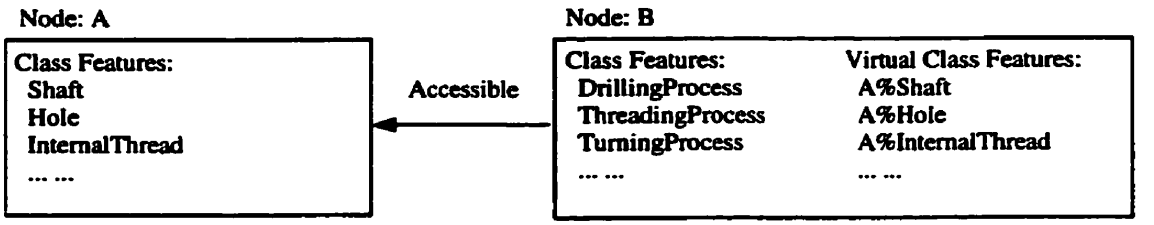


Figure 3.11 Virtual Class Features

Virtual class features in remote nodes can be displayed in the Class Feature Browser, by executing the Display Virtual Class Features in the menu of feature list view as shown in Figure 3.12. The Class Feature Browser is a previously developed browser, but more functions, such as displaying virtual class features, are added. Detailed descriptions about this browser are given in [Yadav 1999].

3.3.1.2 Virtual Instance Features

Virtual instance features are instance features preserved in the accessible remote nodes. They are part of the databases for modeling specific product development activities. Similar to virtual class features, a virtual instance feature is named in the following format:

<node name>%<instance feature name>

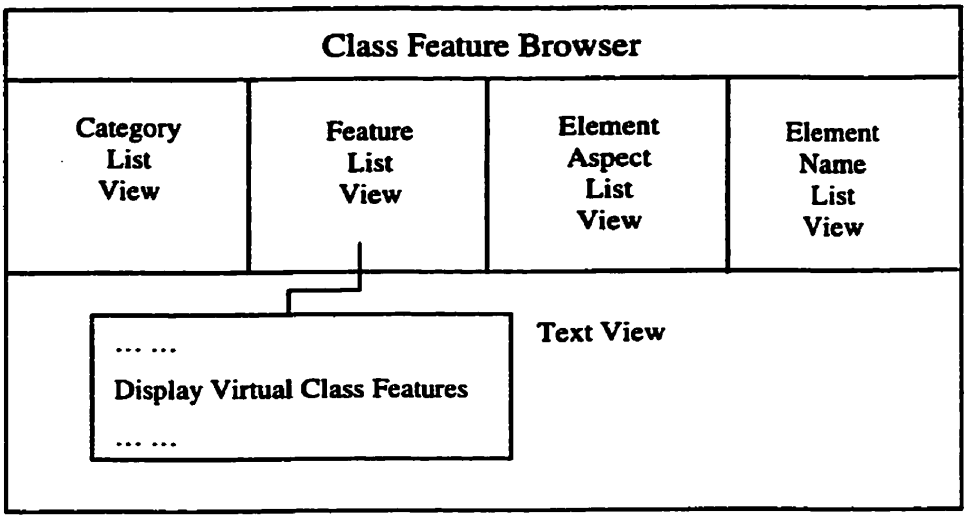


Figure 3.12 The Views in the Class Feature Browser

For example, an instance feature **pulley1** preserved in node **A** is described as **A%pulley1** in node **B** (Figure 3.13). The virtual instance features at remote nodes accessible from the local node are considered as part of the database at the local node. By associating remote databases with the local database for modeling a product, the different product development processes can be integrated into the same environment. Such integration is necessary for implementing concurrent design using distributed databases.

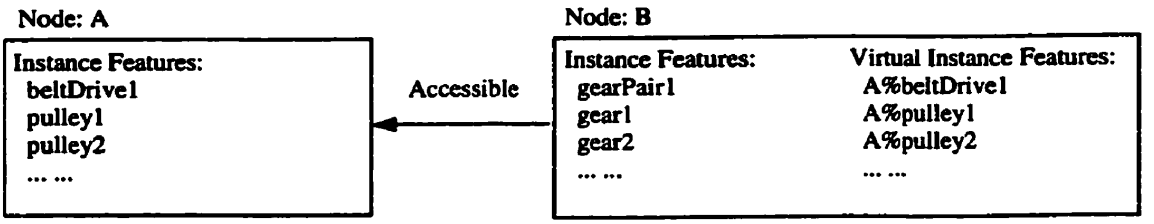


Figure 3.13 Virtual Instance Features

In Figure 3.13, node **A** has instance features for modeling a pulley-belt drive mechanism and node **B** has instance features for modeling a gear pair mechanism. At node **B**, if a motion transfer mechanism consisting of both a gear pair and a pulley-belt drive needs to be modeled, then the true instance features **gearPair1**, **gear1**, **gear2**, etc.

and the virtual instance features A%beltDrive1, A%pulley1, A%pulley2, etc. are associated by defining their relations among the databases preserved in these two nodes.

All virtual instance features in the accessible remote nodes, including their attributes, attribute relations, feature relations, etc., can be displayed in the Instance Feature Browser. The configuration of this previously developed browser, as shown in Figure 3.14, remains unchanged. The menus in this browser are modified to accommodate more commands such as Display Virtual Instance Features that is used to view the instance features preserved in all accessible remote nodes. The attributes of the virtual instance features are described in the format of

<attribute name>[<virtual instance feature name>]

For example, the attribute length of instance feature shaft1 in a remote node called ShaftDesign is described at the local node by:

length[ShaftDesign%shaft1]

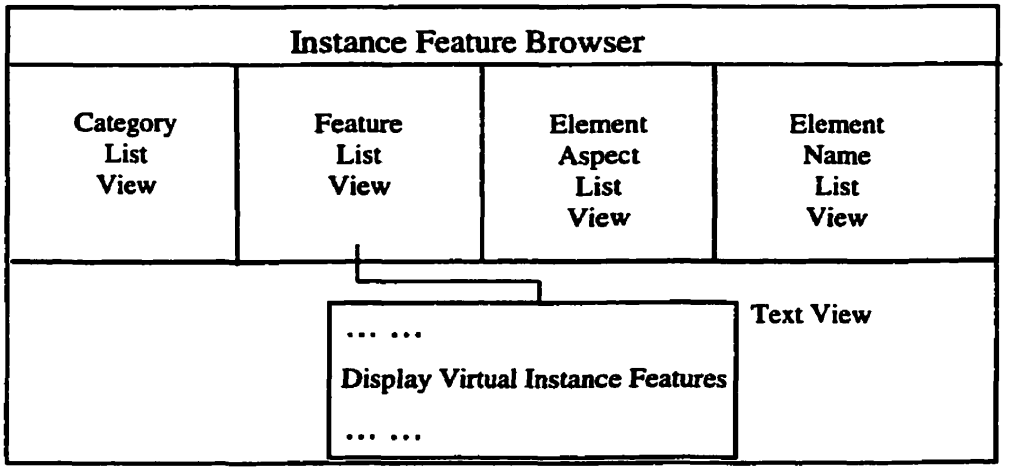


Figure 3.14 The Views in the Instance Feature Browser

3.3.1.3 Generation of Instance Features from Virtual Class Features

In this distributed database and knowledge base modeling system, a virtual class feature can be used to generate a true instance feature at the local node. The instance features generated from the virtual class features are treated in the same manner as those

generated from the true class features. In other words, they are truly part of the database in the modeling processes.

A local true instance feature is an object that contains data and operations that are necessary for modeling the products or the product development activities. In this feature-based modeling system, the class features are translated into Smalltalk classes, so the instance features are actually instances of Smalltalk classes. Therefore, the operations, such as instance methods defined in class features, can be inherited by the instance features.

For generating a true instance feature using a virtual class feature preserved in a remote node, a special class feature, called VirtualClassFeature, is predefined in the system. An instance feature of this class feature is actually an empty instance with the same structure as a regular true instance feature. This instance feature is then filled with descriptions from the corresponding class feature and its super-class features in the remote node. The descriptions are obtained by sending a message from the local node to the remote node. Then the descriptions in that class, including those inherited descriptions from super-classes, are copied to the empty instance feature at the local node. In this way, all the descriptions defined in the virtual class feature and its super-class features are inherited into this generated true instance feature automatically. During the product development process, when an instance feature requires the information from its virtual class feature, a message is sent from the local node (client) to the remote node (server) using the client-server communication architecture.

Figure 3.15 illustrates the process of generating a true instance feature using a virtual class feature. The class feature Gear in node Design1 is used to generate a true instance feature gear1 in node Design2. First, the built-in class feature VirtualClassFeature is selected to generate an empty instance feature gear1 which has the same structure as those regular instance features. In other words, gear1 has all built-in aspects such as Element-features, Attributes, Attribute-relations, Feature-relations, etc., but there are no elements or descriptions in these aspects at this moment. When VirtualClassFeature is used to generate an instance feature, the class feature name

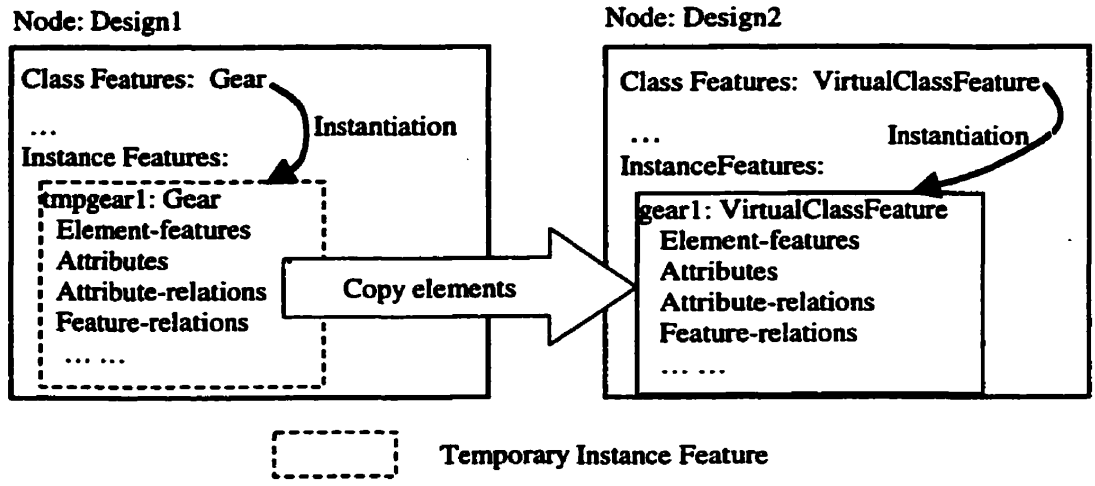


Figure 3.15 Generation of a True Instance Feature from a Virtual Class Feature

Gear and the node name Design1 are requested from the user and recorded as instance variable values of the newly generated instance feature. A message is then sent to the remote node Design1 to ask that node to generate a temporary instance feature tmpgear1 from class feature Gear. After tmpgear1 is generated, the elements and descriptions in all the aspects are copied back to the empty instance feature gear1 in node Design2. Since tmpgear1 inherits all descriptions from class feature Gear and its super-class features, the instance feature gear1, a copy of tmpgear1, in node Design2 inherits all descriptions from virtual class feature Design1%Gear and its super-class features. The temporary instance feature tmpgear1 is then removed from node Design1. Except for the entering of the node name and the class feature name, this process of generating a true instance feature from a virtual class feature is conducted automatically.

3.3.2 Modeling Database Relations

One of the objectives of this research is to associate the different databases at different locations into an integrated environment. The distributed computers, which contain databases for modeling different product components or different product development activities, are connected together through the Internet. The generic relations of the databases, i.e., the relations among the true instance features and the virtual

instance features, also need to be defined so that these distributed databases are integrated effectively.

In this research, the relation between a virtual instance feature and a true instance feature is modeled by defining the virtual instance feature as an element feature of the true instance feature. The relation can be created at two different levels: class feature level and instance feature level, respectively.

3.3.2.1 Modeling Database Relations at Class Feature Level

Creation of a relation between a virtual feature and a true feature at class feature level is conducted by introducing a virtual element-feature. A virtual element feature in a class feature is defined by an element-feature variable and its class type called VirtualInstanceFeature, as shown in Figure 3.16. VirtualInstanceFeature is a built-in class feature used specially for modeling relations among true features and virtual features.

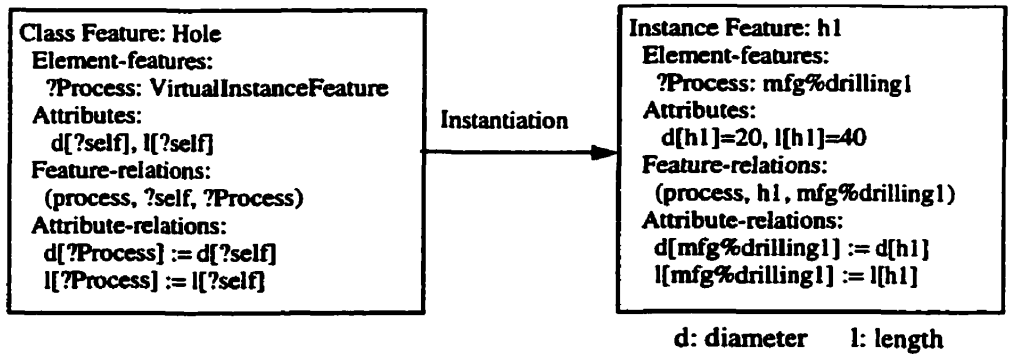


Figure 3.16 Relations among True and Virtual Instance Features Defined at Class Feature Level

In the example shown in Figure 3.16, the virtual instance feature is associated with an element feature variable ?Process which represents a manufacturing process feature. The diameter attribute, d, and the length attribute, l, of this virtual feature are calculated using the attributes of the true feature. When the class feature Hole is used to generate the instance feature h1, the user is asked to enter the virtual instance feature name, including the node name (e.g., mfg) and the instance feature name (e.g., drilling1). All the variables

related to the virtual element feature in the class feature definition should be replaced by the actually created virtual instance feature name as shown in Figure 3.16. When the attribute values at the current node are changed, the relevant attribute values at the remote nodes should also be updated using these relations. Details regarding the maintenance of the data dependency relations in the distributed database modeling system will be discussed in Section 3.3.3.

3.3.2.2 Modeling Database Relations at Instance Feature Level

Modeling of relations among true and virtual instance features at instance feature level is conducted by adding an element-feature, representing a virtual instance feature, to the current instance feature. In the example shown in Figure 3.17, an instance feature h1 with two attributes is first created. The virtual instance feature, mfg%drilling1, is then added to the instance feature h1 as an element-feature. Subsequently, feature relations and attribute relations are added. So the relations between true instance feature h1 and virtual instance feature mfg%drilling1 are established.

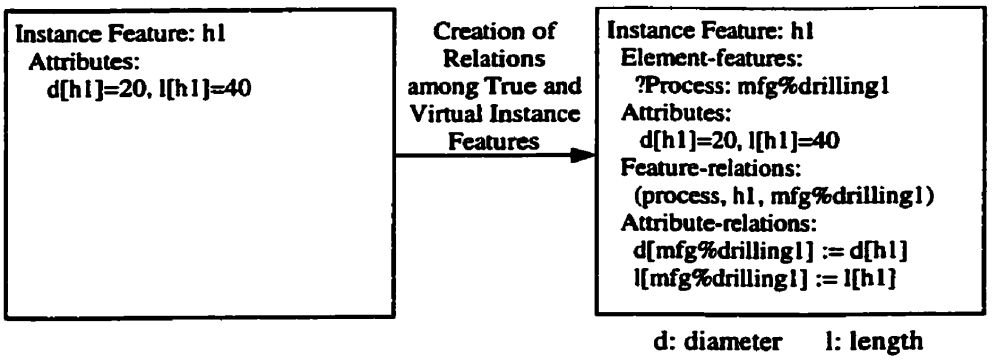


Figure 3.17 Relations among True and Virtual Instance Features Defined at Instance Feature Level

3.3.3 Maintenance of Dependency Relations among Distributed Data

In the product development process using concurrent design methodology, the databases at different locations are used to model the different development life-cycle aspects of the same product. The product development life-cycle covers marketing,

design, manufacturing, maintenance, and so on. The product design, modeled by a feature-based database in this research, should be dynamically evaluated by the performance of this design in down-stream life-cycle phases. In other words, any change of the data in one product life-cycle aspect should be propagated to other aspects automatically according to the data relations defined in the databases.

In the example shown in Figure 3.18, a relation has been defined between instance feature shaft1 and virtual instance feature ShaftMfg%shaftProcess1. During concurrent design process, if the value of length attribute l[shaft1] in node ShaftDesign is modified, the attribute relation

l[ShaftMfg%shaftProcess1] := l[shaft1]

defined in ShaftDesign will lead to the change of the attribute l[shaftProcess1]'s value in node ShaftMfg. The value of attribute cost[shaftProcess1] is automatically modified based upon the updated value of l[shaftProcess1]. The value of cost[shaftProcess1] is then propagated back to node ShaftDesign using attribute relation

mfgCost[ShaftDesign%shaft1] := cost[shaftProcess1]

to update mfgCost[shaft1]. The value of mfgCost[shaft1] can be used as one of the measures to evaluate the manufacturability of the shaft design.

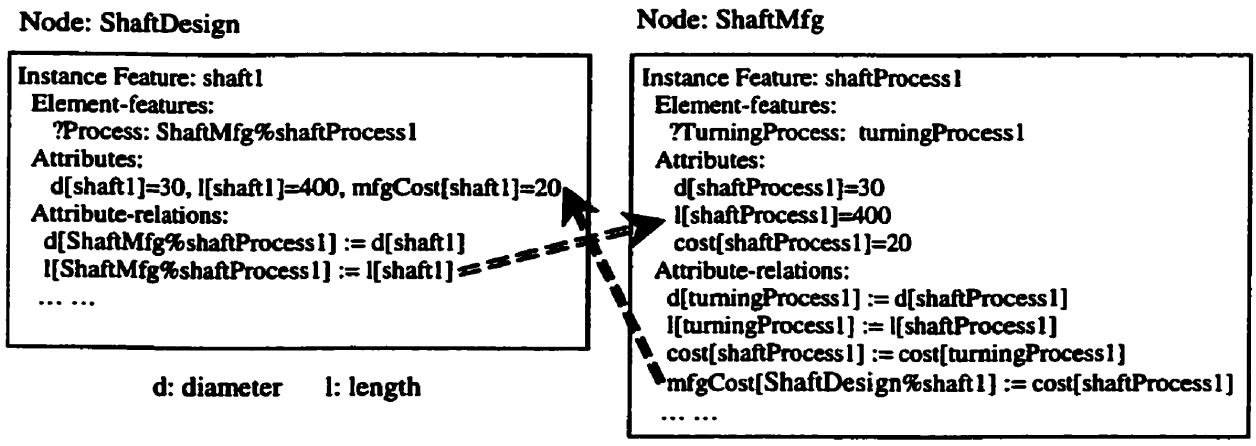


Figure 3.18 Attribute Propagation Process

In this research, a mechanism to maintain the dependency relations of distributed attributes has been developed.

3.3.3.1 An Algorithm for Maintaining Distributed Data Dependency Relations

Maintenance of the attribute dependency relations within one Internet node is carried out using the algorithm introduced in Section 2.3. When the attribute values of virtual instance features are changed, the attribute dependency relation maintenance mechanisms in these remote Internet nodes are then activated to propagate the change using the attribute relations defined in these nodes. This process is carried out continuously until all the relevant attributes in these nodes are updated. The propagation of the attribute value changes is started from a selected Internet node. The algorithm for calculating attribute change propagation at one Internet node to keep the consistency of the distributed database is formulated in the following steps.

Step 1: Identify all the attributes, including true attributes and virtual attributes, whose values have been changed at the current node. Use the attribute dependency relation maintenance mechanism introduced in Section 2.3 to update the change of attribute values using the attribute relations defined at the current node.

Step 2: Obtain all the attributes of the virtual instance features whose values have been changed in Step 1. When such attributes can be found, jump to Step 3. When no such attributes exist, if the current node is the one selected for starting the calculation of the attribute change propagation, the calculation should be terminated. If the calculation is initiated from another node, return a nil value to this node to resume calculation at this remote node and terminate calculation at the local node.

Step 3: Group all the changed virtual attributes according to their nodes. When no changed virtual attribute is in the node from which the execution is initiated, return a nil value to this node to resume the calculation at this node. For each remote node, take the following steps:

(a) If the execution of the current attribute dependency relation maintenance mechanism is initiated from that node, the collection of the changed virtual attributes should be sent back to this node as the return value to resume the execution of the attribute dependency relation maintenance mechanism at that node.

(b) If the node is not the one from which the execution is initiated, send a message to the node to inform the changed attributes and activate the calculation using the attribute dependency relation maintenance mechanism at this remote node. Suspend execution of the attribute dependency relation maintenance mechanism at the local node to wait for the execution result from the remote node.

Step 4: Go to Step 1.

The attribute change propagation calculation is started from one node. Since each node is associated with an attribute dependency relation maintenance mechanism, execution of these mechanisms at these nodes can be conducted simultaneously. Therefore, the distributed attribute dependency relation maintenance mechanism has the nature of concurrent parallel computing. When no contradictory relations exists in the distributed attribute relation network, the consistency of the distributed attribute relations can be maintained.

In this algorithm, a node reacts firstly to the message received first. When a large number of nodes and large amount of information are involved in the product development process, a more robust coordination mechanism is required to handle the messages and coordinate the actions among the distributed Internet nodes. An intelligent agent can be used for this task. The coordination and cooperation of the distributed Internet nodes should be improved in future studies.

3.3.3.2 An Example of Attribute Propagation Process

This algorithm to maintain the consistency of the distributed database is illustrated using an example shown in Figure 3.19. In this example, the attribute a1 in node A is changed at the very beginning. The calculation is conducted through the following steps:

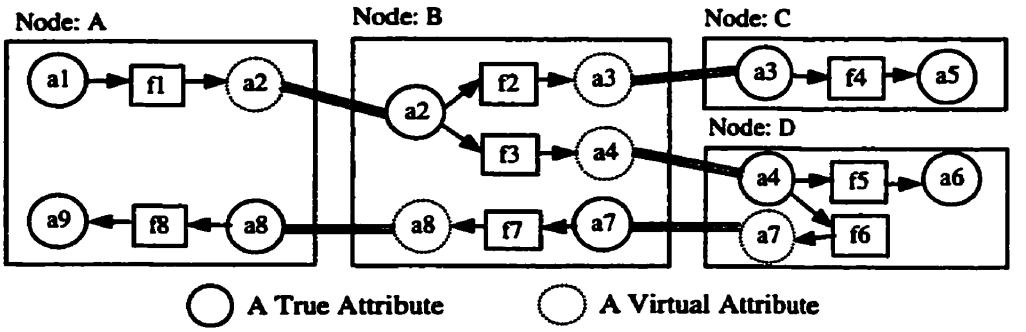


Figure 3.19 Maintenance of the Distributed Database

1. At node A, since a1 is the attribute whose value has been changed, function f1 is then activated to update the change to the attribute a2. Because a2 is a virtual attribute preserved in node B, a message is then sent to node B to activate the calculation at node B. The node A changes to the mode to wait for the execution result from node B.

2. At node B, since a2 is the attribute whose value has been changed, functions f2 and f3 are then activated to update the change to attributes a3 and a4. Because a3 and a4 are virtual attributes preserved in node C and node D respectively, messages are then sent to node C and node D to activate the calculation at these two nodes. The node B changes to the mode to wait for the execution results from node C and node D. A message is also sent to node A with a nil value.

3. At node A, a message nil is received from node B and the execution at node A is terminated.

4. At node C, since a3 is the attribute whose value has been changed, function f4 is then activated to update the change to attribute a5. Because no virtual attribute is changed, a message with return value nil is sent to node B and the calculation at node C is terminated.

5. At node D, since a4 is the attribute whose value has been changed, functions f5 and f6 are then activated to update the change to attributes a6 and a7. Because a7 is a virtual attribute preserved in node B from which the execution is initiated, a message with return value of a7 is sent back to node B and the calculation at node D is terminated.
6. At node B, a message with return value nil is received from node C and a message with return value of a7 is received from node D. Since a7 is the attribute whose value has been changed, function f7 is then activated to update the change to attribute a8. Because a8 is a virtual attribute preserved in node A, a message is then sent to node A to activate the calculation at node A. The node B changes to the mode to wait for the execution result from node A.
7. At node A, since a8 is the attribute whose value has been changed, function f8 is then activated to update the change to attribute a9. Because no virtual attribute is changed, a message with return value nil is sent back to node B and the calculation at node A is terminated.
8. At node B, a message nil is received from node A and the execution at node B is terminated.

3.4 Distributed Knowledge Base Modeling

To improve the efficiency of the product modeling process, a rule-based inference mechanism is employed to help designers to modify the product databases. In the feature-based database and knowledge base modeling system, each instance feature is associated with a number of rule-bases as introduced in Section 2.3 to improve the inference efficiency by considering only partial database and knowledge base. In this research, this idea is extended to the distributed database and knowledge base modeling.

In one node, if a rule-base is selected for the active instance feature, the name of this selected rule-base is then registered in the active instance feature. During the reasoning process, all rules in the selected rule-bases are used in inference. A previously developed

Rule-base Browser is used to define rule-bases [Yadav 1999]. This browser remains unchanged except for adding a command in the menu for displaying virtual rule-bases.

Like product database modeling, product knowledge bases are modeled at different locations. To use knowledge preserved in remote nodes for product development, the concept of virtual rule-bases, i.e., the rule-bases defined in accessible remote nodes, is introduced. A mechanism of distributed inference is also developed to handle the distributed databases.

3.4.1 Virtual Rule-Bases

When a node A is defined to be able to access another node B, all the rule-bases defined in node B are then accessible from node A. The rule-bases defined in accessible remote nodes are called *virtual rule-bases*. A virtual rule-base is described in the following format:

<node name>%<rule-base name>

For example, a rule-base GearDesignRules in node B is described in node A by:

B%GearDesignRules

During the product development process, virtual rule-bases can also be selected for reasoning together with the selected true rule-bases at the local node. This idea is illustrated in Figure 3.20.

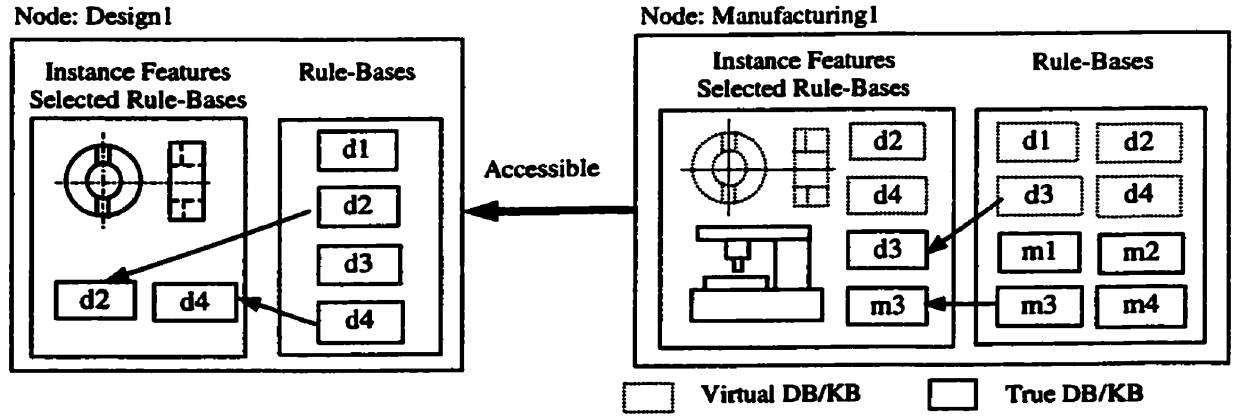


Figure 3.20 Virtual Rule-Bases

Generally the virtual rule-bases are those preserved in the remote nodes and can be selected to access the data in this active instance feature at the local node through knowledge-based inference. When a virtual rule-base is selected, all the rules in this rule-base are then copied to and registered in the active instance feature of the local node.

During the process of product development through knowledge-based reasoning, both the true rule-bases and the virtual rule-bases selected for the active instance features are used as the knowledge to access the data represented by active instance features at the local node.

In the example shown in Figure 3.20, the node Design1 is accessible from the node Manufacturing1. The rule-bases, d1, d2, d3, and d4 at node Design1 are virtual rule-bases at node Manufacturing1. In node Manufacturing1, the virtual rule-base, d3, and true rule-base m3, are selected to join the reasoning at the local node.

The virtual rule-base modeling mechanism allows engineers to improve the efficiency of product development through sharing distributed knowledge bases. The characteristics of this distributed knowledge base modeling approach are summarized into the following two aspects.

(1) By using virtual rule-bases at the local node, the standard knowledge at a remote node can be "borrowed" to the local node. This mechanism allows the different knowledge bases to be modeled at different places. During the process of product development, when certain types of knowledge are required, the system then identifies the locations of the required knowledge and introduces the required knowledge for the product currently under development at the local node. This mechanism can also solve the problems of knowledge representation redundancy due to the fact that the same knowledge is described in many places. The introduced knowledge at the local node is dynamic in nature, i.e., when the accessibility relations among the nodes are changed, the virtual rule-bases at the local node are then removed.

(2) By using virtual rule-bases, all the knowledge used in different product development phases can be integrated into the same environment. The rule-base used by a down-steam phase of product development can be integrated into the current phase by selecting this virtual rule-base. This approach can result in better design databases in terms of the performance of the design in down-steam development life-cycle. When the knowledge in different knowledge bases is in conflict, the conflict can be resolved either by changing the introduced knowledge bases at different locations, or by modifying the accessibility relations among the Internet nodes to change the product development alternatives.

3.4.2 Selection of Virtual Rule-B ases

The selection of rule-bases, including true rule-bases and virtual rule-bases, is conducted using the **Rule-Base Selection Browser** shown in Figure 3.21 and 3.22. Rule-base selection is a process of selecting relevant reasoning rules for the active

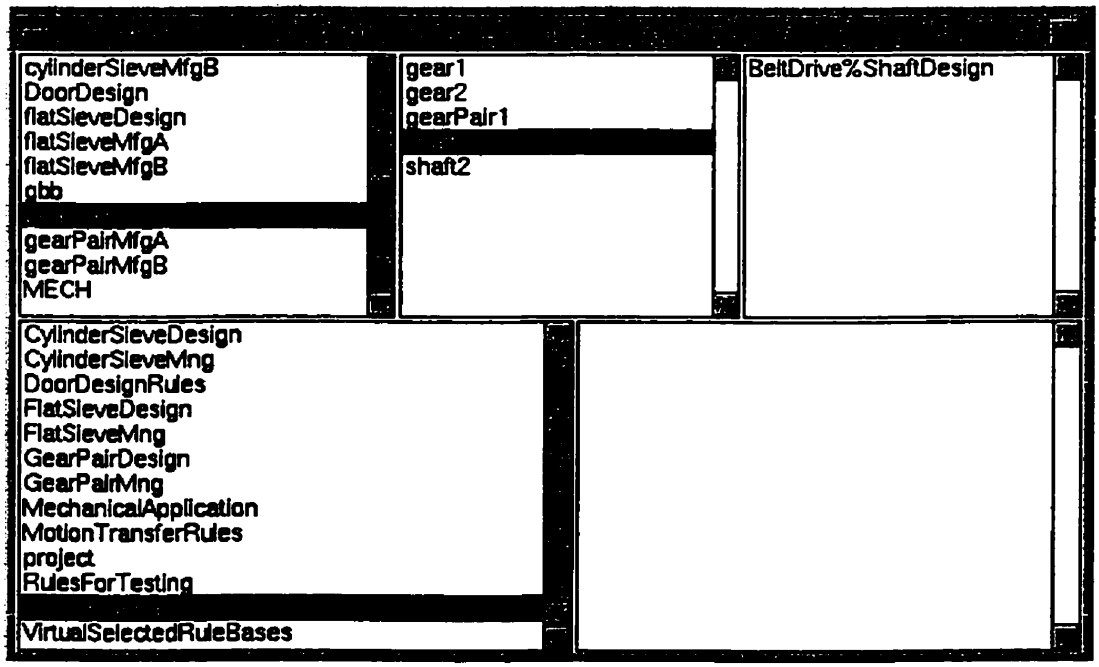


Figure 3.21 A Snapshot of the Rule-Base Selection Browser

instance features. Therefore this browser contains information from both **Instance Feature Browser** and **Rule-Base Browser**. In this browser, the **Instance Feature Category List View** and the **Instance Feature List View** are used to display instance feature categories and instance features, while **Rule-Base Category List View** and **Rule-Base List View** are used to display rule-base categories and rule-bases. The **Selected Rule-Base List View** is used to display the selected rule-bases for the instance feature highlighted in the **Instance Feature List View**. Two commands, **Select** and **Remove**, are implemented in the **Rule-Base List View** and **Selected Rule-Base List View** respectively. The command **Select** is used to add the rule-base highlighted in the **Rule-Base List View** to the active instance feature. The command **Remove** is used to delete a rule-base highlighted in the **Selected Rule-Base List View** from the current active instance feature.

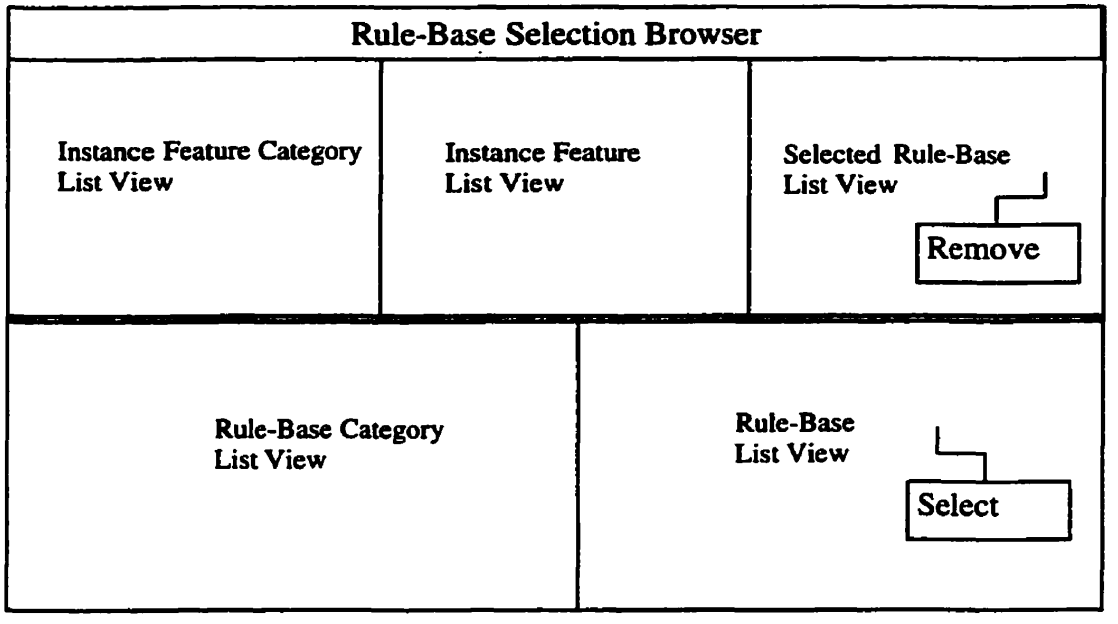


Figure 3.22 Configuration of the Rule-Base Selection Browser

For selecting virtual rule-bases, a special category for the rule-bases, namely **VirtualRuleBases**, is defined in the **Rule-Base Category List View**. When the **VirtualRuleBases** category is highlighted for rule-base selection, the system will ask for

the Internet node name and the rule-base name. After that information is correctly supplied, the virtual rule-base is selected and its name is displayed in the **Selected Rule-Base List View.**

The virtual rule-bases selected are treated the same as those true rule-bases selected from the local knowledge base library. In this research project, the selection of relevant rule-bases for product development is conducted manually. Agent-assisted rule-base selection should be studied in the future to improve the efficiency of knowledge selection.

3.4.3 Reasoning with Distributed Rule-Bases

In feature-based product modeling system, development of a product can be conducted through rule-based reasoning. Since the different product aspects are modeled in different Internet nodes, rule-based reasoning is also conducted in these different places.

In each Internet node, only partial databases and knowledge bases are selected for knowledge-based reasoning to improve product development efficiency. The selected databases are represented by the active instance features, including attributes, qualitative relations among instance features, and quantitative relations among attributes in distributed accessible nodes. The selected knowledge bases are represented by both the selected true rule-bases and the selected virtual rule-bases. These rule-bases are registered with the active instance features of the accessible nodes.

In the rule-based inference, the condition parts of all the rules in both true rule-bases and virtual rule-bases are matched with the selected partial database. Among all the rules whose condition parts have been satisfied, the best rule is selected according to the conflict resolution strategy, and the result part of the best rule is then executed. In this research, the first matched rule is considered as the best rule in rule-based reasoning. Matching of the condition parts and execution of the result parts for a rule in a virtual rule-base are conducted in the same manner as those of true rule-bases introduced in Section 2.3.2.

Since the rule-based reasoning at one node can result in the changes of the virtual data at the remote nodes, the executions of the rule-based reasoning mechanisms at these remote nodes is then required to update the changes. Since the rule-based reasoning in the different nodes can be conducted simultaneously, the distributed knowledge-based reasoning mechanism has the nature of concurrent parallel computing.

The algorithm for executing the rule-based inference mechanism at one Internet node during distributed knowledge-based reasoning is formulated in the following steps.

Step 1: Use all the rules in the selected the true rule-bases and virtual rule-bases to access the database represented by active instance features through matching the condition parts and executing the result parts of these rules.

Step 2: Obtain all the virtual data that have been changed in Step 1. When such data can be found, jump to Step 3. When no such data exist, if the current node is the one selected for starting the rule-based reasoning, the inference should be terminated. If the reasoning is initiated from another node, return a nil value to this node to resume inference at this remote node and terminate inference at the local node.

Step 3: Group all the changed virtual data according to their nodes. When no changed virtual data is in the node from which the inference is initiated, return a nil value to this node to resume inference at this node. For each remote node, do the following steps:

(a) If the execution of the current rule-based reasoning is initiated from that node, send the collection of the changed virtual data back to this node as the return value to resume the execution of the rule-based inference at that node.

(b) If the node is not the one from which the execution of the rule-based inference is initiated, send a message to the node to inform the changed data and activate the rule-based reasoning at this remote node. Suspend

execution of the rule-based inference at the local node to wait for the inference result from the remote node.

Step 4: If the data preserved at the current node are changed, due to the inference conducted at remote nodes, go to Step 1. Otherwise, terminate the execution of the rule-based reasoning.

This distributed knowledge-based inference algorithm is very similar to the one introduced in Section 3.3.3 for maintaining the dependency relations among the distributed attributes. Since knowledge-based reasoning can result in the change of the attribute values, propagation of the attribute change is then required. The attribute change propagation can further change the product database, thus resulting in change of the conditions for rule matching, and the rule-based reasoning is then required again.

3.5 Summary

This chapter presents a detailed discussion on issues in modeling a feature-based distributed database and knowledge base for concurrent design of engineering products. To associate geographically distributed product development activities, modeled by feature-based databases and knowledge bases, the Internet is employed as the media for connecting the computers used for modeling these databases and knowledge bases. Information flows among different product development activities that are modeled by Internet nodes are realized through the socket-based client–server communication architecture.

To use the remote databases and knowledge bases at a local site, the concepts of virtual features and virtual rule-bases are introduced. The virtual databases and knowledge bases are physically located at remote locations but accessible from the local location. A virtual class feature can be used to generate a true instance feature at the local node. Virtual instance features are considered as part of the databases required for modeling product development processes. Virtual rule-bases can be selected for reasoning together with the true rule-bases selected at local node. The relations among virtual instance features and true instance features are also modeled.

To implement concurrent design, mechanisms for distributed data dependency relation maintenance and distributed inference are developed. The mechanism for distributed data dependency relation maintenance serves as the engine to propagate data changes among the Internet nodes defined. This mechanism can give feedback from the down-stream development processes if the design data are modified. The mechanism for distributed inference helps designers to generate and modify geographically distributed product development databases to improve product development efficiency.

CHAPTER 4

CONCURRENT DESIGN BASED UPON DISTRIBUTED DATABASE AND KNOWLEDGE BASE MODELING

This chapter introduces the development of a product concurrent design system based on the distributed database and knowledge base modeling approach described in chapter 3. Following an introduction, Section 4.2 discusses the methods of modeling product realization processes for concurrent design, including modeling of relations among Internet nodes and representation of product realization process alternatives. Section 4.3 introduces methods for identifying the optimal solution from all feasible product realization process alternatives. Two methods are introduced: the exhaustive method and the Genetic Programming (GP) method. The optimal parameter values for each alternative are identified using a global optimization method called Particle Swarm Optimization (PSO). Section 4.4 introduces this method.

4.1 Introduction

Concurrent design is a methodology in which the related down-stream product development processes are considered concurrently at the design stage [Hyeon et al. 1993]. To apply concurrent design method in product development using computer-based systems, the design model and related down-stream development process models must be integrated to ensure mutual information flows. When the databases used for modeling the product development processes are geographically distributed at different locations, the distributed database and knowledge base modeling system introduced in Chapter 3 provides an effective technique for integrating the distributed product development models.

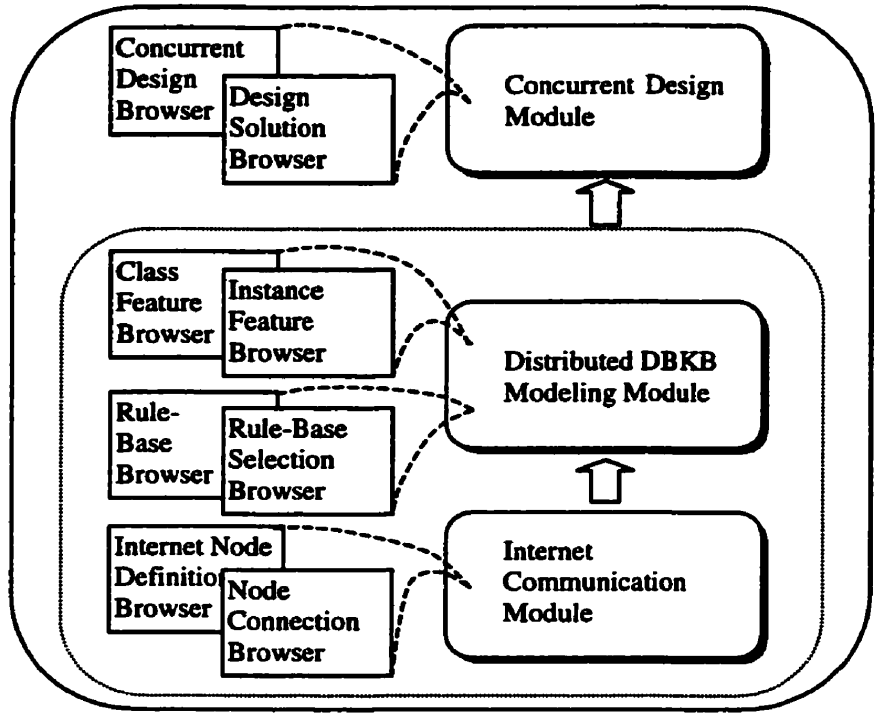
When the distributed databases are integrated, product concurrent design can then be achieved by evaluating the design candidates using the down-stream product development process models. The distributed database and knowledge base modeling approach introduced in Chapter 3 provides a framework for modeling the different

product development processes and their relations. Based on this approach, the design parameters can be optimized in terms of the product performance in down-stream development life-cycle phases. The product concurrent design can be conducted by adjusting the design parameter values and evaluating the design using the feedback from the related down-stream product development models. To improve the efficiency of product development, a global optimization method is employed to automate the concurrent design process in this research.

In today's global product development environment, alternative processes can be employed for product development at one life-cycle phase such as design and manufacturing. For example, there may be two or three Internet nodes that can handle gear manufacturing independently. By selecting different Internet nodes, different product realization processes can be obtained. Selection of relevant Internet nodes, representing different product development life-cycle models, for identifying the optimal alternative for product development, is one of the issues of the concurrent design to be discussed in this research. If the number of involved Internet nodes is small, the best concurrent design solution can be determined by comparing all feasible alternatives. If the number of involved Internet nodes is large, the optimal concurrent design solution should be identified using the optimization method.

Figure 4.1 shows the architecture of the concurrent design system developed in this research. The concurrent design module was developed based upon the distributed database and knowledge base modeling approach introduced in Chapter 3. This module is accessed by the Concurrent Design Browser and the Design Solution Browser. The distributed database and the knowledge base modeling system is composed of the Internet communication module and distributed database and knowledge base modeling module.

The following sections will discuss the methods for modeling product realization process alternatives and identifying the optimal product realization process alternative. The optimization method for identifying the optimal parameter values will also be introduced.



DBKB: Database and Knowledge Base

Figure 4.1 Architecture of the Concurrent Design System

4.2 Modeling of Product Realization Process Alternatives

A product realization process alternative is a route of product evolution from design to down-stream development processes. During concurrent design, the down-stream product development processes are considered concurrently to improve the performance of the design in down-stream life-cycle phases. This section discusses issues in modeling product realization processes.

4.2.1 The Relations among Internet Nodes

The product development activities, such as design and manufacturing, are modeled by features in this research. In the Internet-based concurrent design system, the activities for different product development stages are modeled using the features distributed at different Internet nodes. So the selection of proper databases and knowledge bases for product development can be regarded as the selection of suitable Internet nodes that are involved in the concurrent design. Therefore, an Internet node can be used to represent

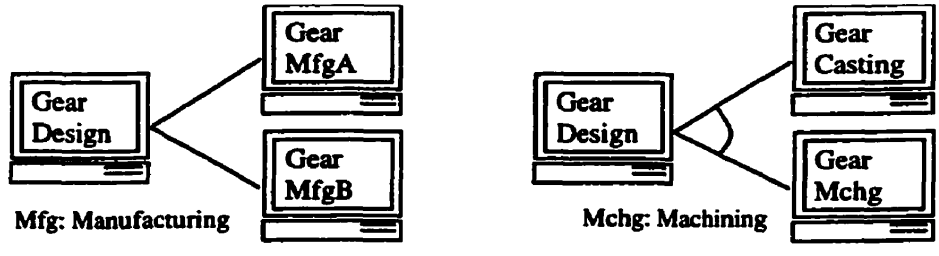
the development activity in a certain phase of the product development life-cycle. For example, an Internet node may represent design activity and another node may represent manufacturing activity, and so on.

4.2.1.1 Logical Relations among Internet Nodes

In this research, the logical relations among Internet nodes that represent the development activities at different stages of the product development life-cycle are defined as node–sub-nodes relations. These relations follow the sequence of activities in the product development life-cycle. For example, a manufacturing node is a sub-node of a design node, since the manufacturing activity usually takes place after design activity.

The relation among the sub-nodes of an Internet node is either an AND relation or an OR relation. The AND relation means that all these sub-nodes, representing sub-processes, are required for modeling the development activity at certain stage of the product development life-cycle. When an OR relation is defined, only one of the sub-nodes is needed for modeling the required product development activity.

In Figure 4.2 (a), Gear Design represents a design node and Gear MfgA and Gear MfgB represent two manufacturing nodes. Gear MfgA and Gear MfgB, the sub-nodes of Gear Design, have an OR relation, which means that either Gear MfgA or Gear MfgB is required for modeling the manufacturing process of the gear. Figure 4.2 (b) shows two sub-nodes with an AND relation. Each of the two nodes handles part of the manufacturing processes: the Gear Casting node handles the casting process and the



(a) Two Sub-Nodes with an OR Relation (b) Two Sub-Nodes with an AND Relation

Figure 4.2 Internet Node Relations

Gear Mchg node handles the machining processes. The gear can be produced using the data and knowledge in both Gear Casting node and Gear Mchg node.

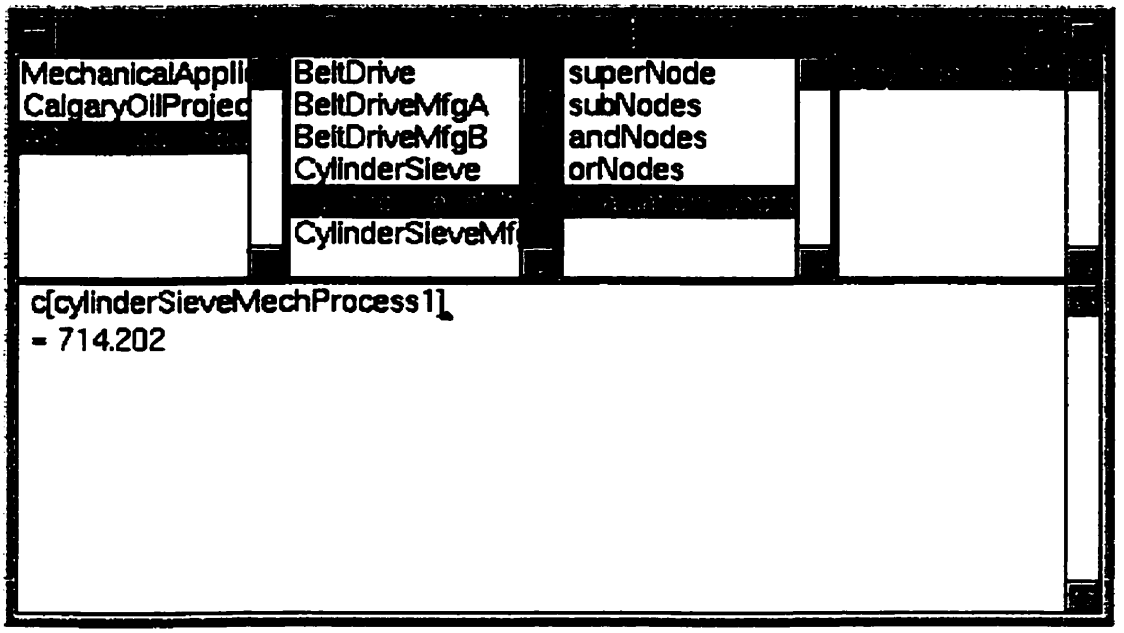


Figure 4.3 A Snapshot of the Concurrent Design Browser

4.2.1.2 Creation of Internet Node Relations

To create Internet node relations for identifying the product realization process alternatives, an interface called Concurrent Design Browser has been developed. A snapshot of the Concurrent Design Browser is shown in Figure 4.3. The configuration of the browser is shown in Figure 4.4.

All the Internet nodes are grouped into different categories. The categories are defined in the category list view. When a category is selected, all Internet node names defined in that category are listed in the node name list view. A new node can be added to a selected category. In the aspect list view, five built-in aspects: superNode, subNodes, andNodes, orNodes, and evaluationFunction, are listed for a selected node. The element list view shows the elements: node names or an evaluation function for the highlighted Internet node. These elements are edited with the use of the text view. For the example shown in Figure 4.5, the node D2 is defined by

Node: D2

superNode: F

subNodes: M3, M4

andNodes: D1

orNodes:

evaluationFunction: <anEvaluationFunction>

After the relations of the involved nodes are defined, the product realization process alternatives can be generated by executing the menu items of the Concurrent Design Browser.

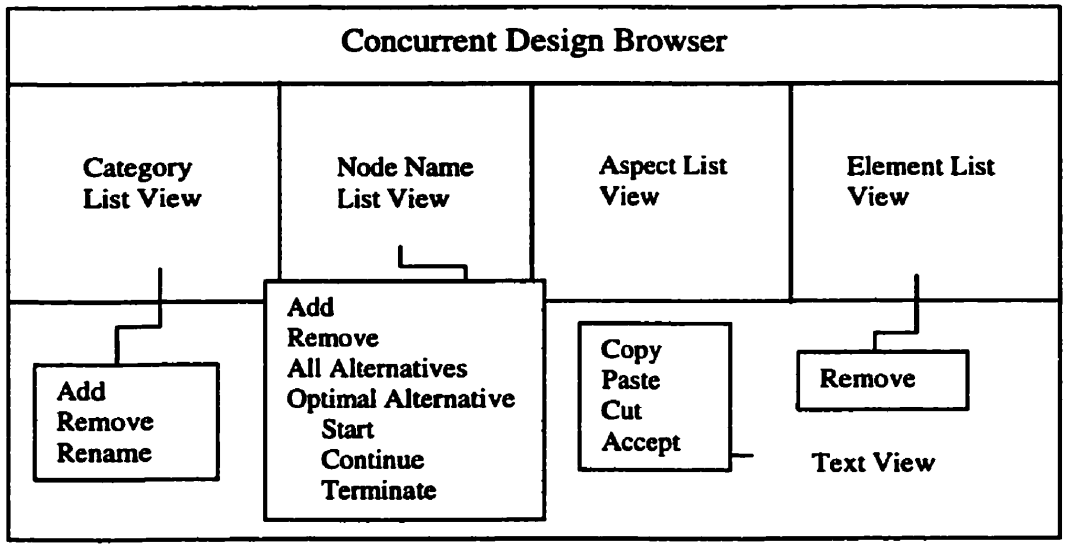


Figure 4.4 Configuration of the Concurrent Design Browser

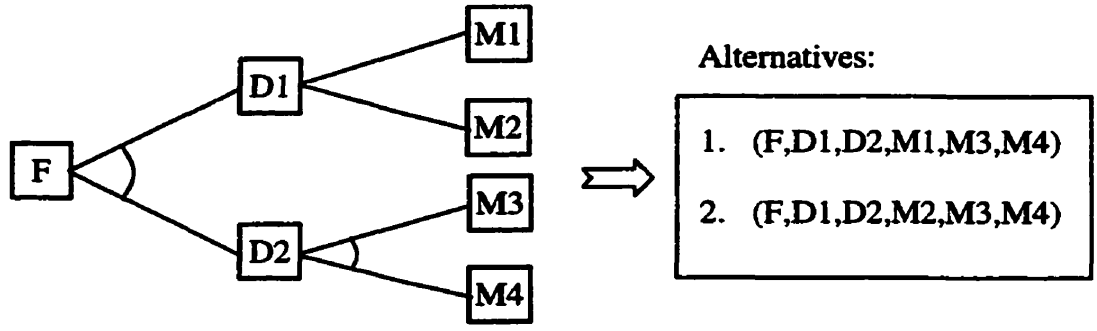
4.2.2 Representation of Product Realization Process Alternatives

After all the relations of the involved Internet nodes are defined using the Concurrent Design Browser, the product realization process alternatives can then be identified.

4.2.2.1 Product Realization Process Alternatives

A product realization process alternative is described by a list of Internet nodes that contain the required databases and knowledge bases for modeling the product

development activities at different stages of the product development life-cycle. For example, Figure 4.5 shows the feasible product realization processes represented by an AND/OR graph with seven Internet nodes. Two product realization process alternatives can be generated from this AND/OR graph. The generated product realization process alternatives are displayed in the **Design Solution Browser** that will be introduced in Section 4.2.2.2.



F: Function D: Design M: Manufacturing

Figure 4.5 An AND/OR Graph for Modeling Product Realization Process Alternatives

4.2.2.2 Display of Product Realization Process Alternatives

The product realization process alternatives generated by the system are displayed in the **Design Solution Browser**. These alternatives are then evaluated and compared with each other to identify the solution that satisfies the design requirements. The **Design Solution Browser** is shown in Figure 4.6 and 4.7.

There are five views in the **Design Solution Browser** as shown in Figure 4.7. The category list view lists the categories defined in the **Design Solution Browser**. The product realization process alternatives are listed in the alternative list view. The data list view displays all the instance feature names preserved in the nodes involved in the product realization process alternative selected in the alternative list view. This view lets users know all the instance features used for modeling the development activities in this product realization process. In the evaluation function list view, evaluation functions for the selected product realization process alternative are listed. An evaluation function in

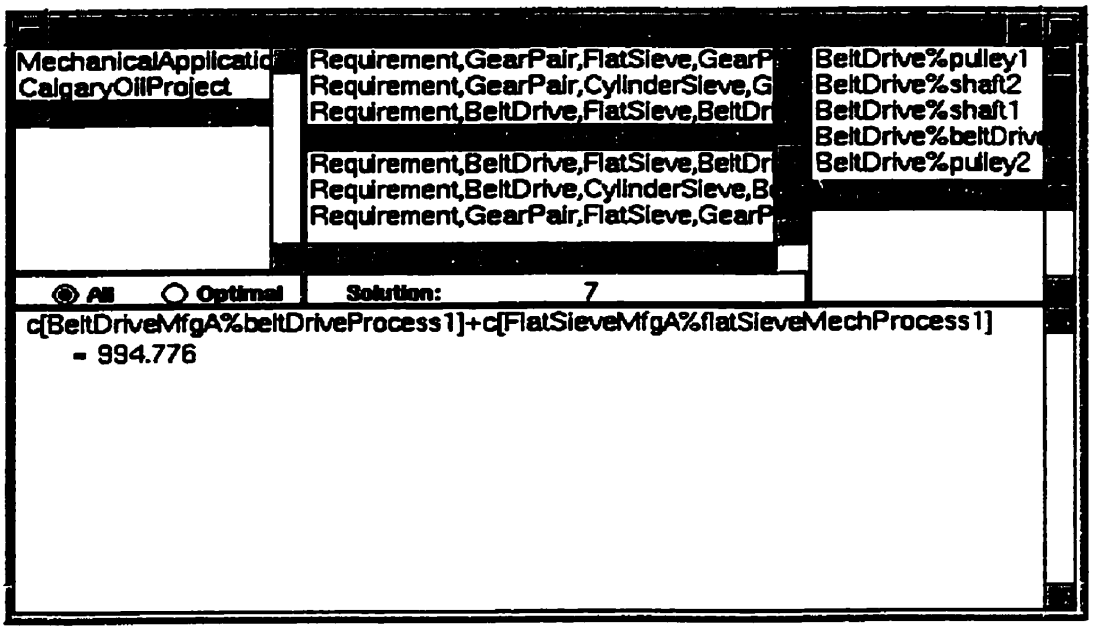


Figure 4.6 A Snapshot of the Design Solution Browser

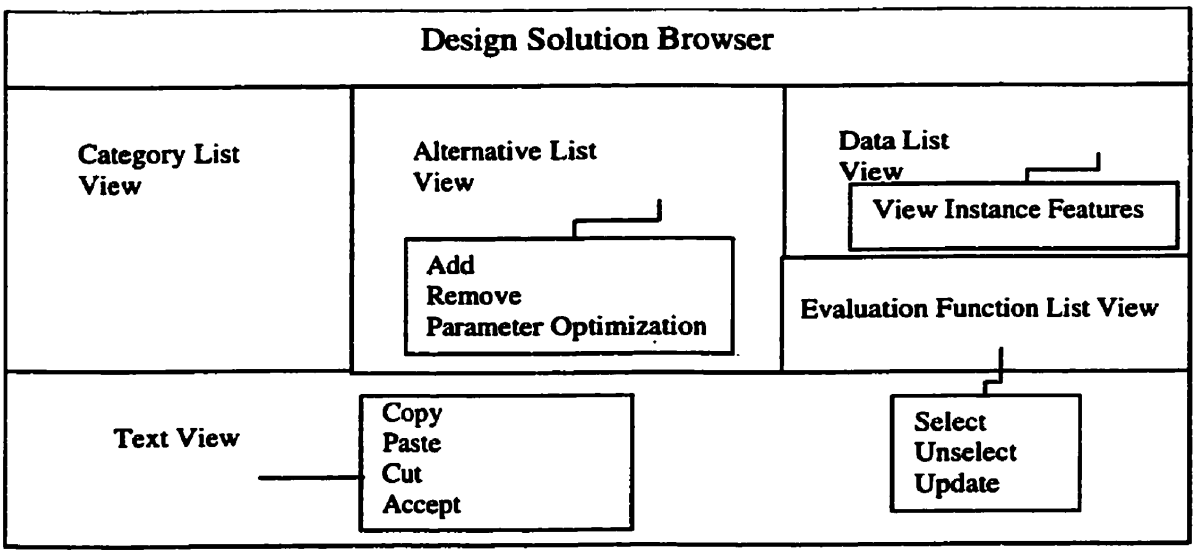


Figure 4.7 Configuration of the Design Solution Browser

this list can be selected to evaluate the highlighted product realization process alternative in the alternative list view. The text view (text editor) is used to edit the evaluation functions. The edited function is saved using the Accept command of the text view menu.

An evaluation function is defined using the attributes of instance features preserved in different Internet nodes. It is used to evaluate the selected product realization process alternative. An evaluation function can be described by $F(\vec{X})$, where $\vec{X}$ is a vector of attributes:

$$\vec{X} = x_1, x_2, \ldots, x_i, \ldots, x_n \qquad (4-1)$$

where x_i is the i-th attribute and n is the total number of attributes used to define this function. For the example shown in Figure 4.5, if the total manufacturing cost is used to evaluate the product realization process alternative (F,D1,D2,M2,M3,M4), the evaluation function takes the following format:

F(cost[M2%m2Process], cost[M3%m3Process], cost[M4%m4Process]) =
cost[M2%m2Process]+cost[M3%m3Process]+ cost[M4%m4Process]

Where *cost* is an attribute name, *m2Process*, *m3Process*, and *m4Process* are instance features for modeling the manufacturing processes in nodes M2, M3 and M4 respectively.

The command Update in the menu of the evaluation function list view brings the updated value of the evaluation function to the text view. Since the attributes used in the evaluation function are distributed at different Internet nodes, messages are sent to these nodes to get the current values of these attributes. The result of the evaluation function is then calculated and displayed.

4.3 Identification of the Optimal Product Realization Process Alternative

The two methods used for identifying the optimal concurrent design solution alternative are (1) the exhaustive method and (2) the Genetic Programming method. When the number of the involved Internet nodes is small, the exhaustive method is used first to generate all possible alternatives. Then the alternatives are evaluated and compared to find the best one. When the number of the involved Internet nodes is large, the Genetic Programming method is used to identify the optimal alternative.

4.3.1 The Exhaustive Method

In the exhaustive method, a list of all product realization process alternatives is first generated automatically. Then the designers can evaluate and compare these alternatives to find the best one.

4.3.1.1 The Algorithm for Generating All Alternatives

After the relations among the involved Internet nodes are defined using the Concurrent Design Browser, all possible alternatives can be generated using the following algorithm:

Step 1: Create an empty collection called alternative collection and an empty list called the node list. Select the root node as the element of the node list. Put the node list into the alternative collection.

Step 2: Pick up a node list, which has unexpanded nodes, from the alternative collection. From this list, pick up a node that is neither a leaf node nor an expanded node. Identify all the sub-nodes of this node.

Step 3: For those sub-nodes with an AND relation, add these nodes into the list. When an OR relation is detected, for each sub-node, a copy of the current list is created and this sub-node is added to the copy. Put these new node lists into the alternative collection and remove the original list.

Step 4: Check whether all the nodes in all the lists are expanded. If no unexpanded node can be found, the expanding process stops. Otherwise, go to Step 2.

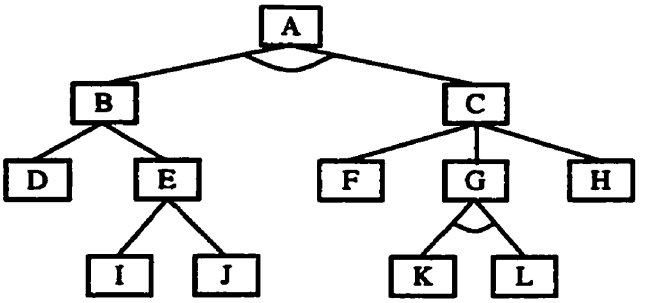
4.3.1.2 An Example of Generating All Alternatives

Suppose that the relations of the involved Internet nodes are defined as shown in Figure 4.8 (a). The product realization process alternatives are generated in the following process:

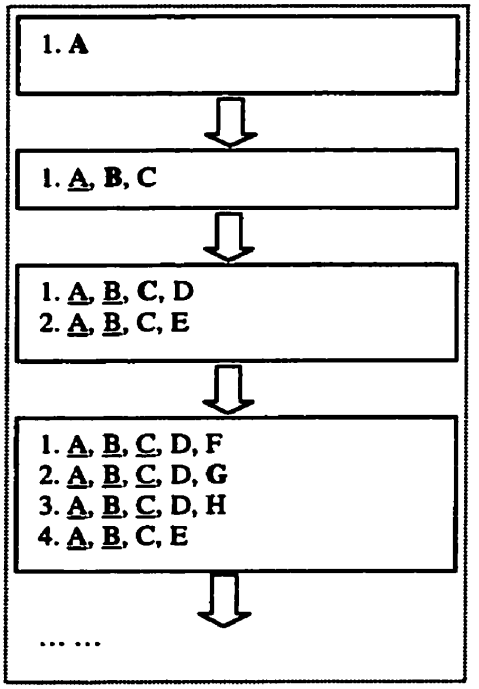
1. The root node A is put into the node list 1. The node list 1 is put into the alternative collection.

2. The node A in node list 1 is picked up for expansion. Since the two sub-nodes of node A, B and C, have an AND relation, these sub-nodes are added to the original node list.

3. Node B in node list 1 is picked up for expansion. Since the two sub-nodes of node B, D and E, have an OR relation, two copies of the original node list 1 are created. The



(a) Product Realization Processes Represented by an AND/OR Graph



(b) Alternative Generation Process

1. A, B, C, D, F
2. A, B, C, D, G, K, L
3. A, B, C, D, H
4. A, B, C, E, F, I
5. A, B, C, E, F, J
6. A, B, C, E, G, I, K, L
7. A, B, C, E, G, J, K, L
8. A, B, C, E, H, I
9. A, B, C, E, H, J

(c) Created 9 Alternatives

Figure 4.8 Generation of All Alternatives

node D and E are added to the two new lists respectively. The original node list 1 is then removed from the alternative collection. Now there are two node lists, node list 1 and 2, in the alternative collection.

4. The node C in node list 1 is picked up for expansion. Node C has three sub-nodes, F, G, and H. Since these sub-nodes have an OR relation, three copies of the original node list 1 are created and the three nodes are added into the three copies respectively. The original node list 1 is replaced with the three new lists.
5. Repeat this process until all nodes in all lists are expanded. Nine alternatives in total are generated, as shown in Figure 4.8 (c).

This process is illustrated in Figure 4.8 (b). In the implemented concurrent design system, the process of generating all possible product realization process alternatives is started by executing the command **All Alternatives** in the menu of node name list view of **Concurrent Design Browser**. The generated alternatives are displayed in the **Design Solution Browser**.

When the number of product realization process alternatives is not large, the designers can evaluate each of them using the defined evaluation function. Then the best alternative can be selected from these alternatives.

4.3.2 The Genetic Programming (GP) Method

If the number of the involved Internet nodes is large, it is impossible for a designer to evaluate all the product realization process alternatives manually. For this reason, Genetic Programming [Koza 1992] is used as an optimization method to identify the optimal alternative.

4.3.2.1 Introduction to Genetic Programming Method

Genetic Programming is an extension of the Genetic Algorithm [Goldberg 1989, Angeline 1994]. As an evolutionary method for search and optimization, Genetic Programming has features suitable for handling more complex problems than the Genetic

Algorithm. The main difference between the two methods is the representation of solutions.

In the Genetic Algorithm, the solution is represented as a string of numbers called chromosomes. A population of such strings evolves generation by generation. These strings are usually fixed-length binary strings and remain in the same length during evolution. One of the limitations of this representation method is that the solutions of some problems are difficult to be coded into fixed-length strings.

In Genetic Programming, the problem solutions are represented by structures such as trees. These structures are manipulated during the evolution process. When a tree is used, the number of branches and the length of each branch change dynamically during the evolution process. Therefore, such solution representation is considered as a dynamic representation.

Though different in problem solution representations, Genetic Programming and Genetic Algorithm share the same principles of evolution through natural selection. Generally there are four steps to solving problems using genetic programming:

Step 1: Generate initial population members randomly. In this research, the members are described as trees. In the population, each individual member, representing a solution to the problem, has a valid structure according to the predefined syntax.

Step 2: Evaluate each individual member based on the fitness predefined according to the problem to be solved. The fitness functions will be introduced at the end of this section.

Step 3: Create a new population using the following operations:

Reproduction: The individuals with better fitness have more chances to be duplicated to the next generation. The individuals to be duplicated are probabilistically selected, based on the fitness of each member, from the population. The number of duplications to be produced depends on how fit the member is.

Crossover: Crossover is also called sexual recombination. Two parental individual members are selected from the population. On each parent, a crossover point is selected randomly. The sub-tree rooted at the selected crossover point can be identified on each parent. Then the sub-tree is removed from its parent and replaced with the sub-tree from the other parent. By switching the two sub-trees, two new offspring are produced for the next generation. After the crossover operation, the syntax defined for the individuals must be maintained.

Mutation: A single individual is randomly selected from the population for mutation. The mutation point is chosen randomly. The sub-tree rooted at that point is replaced with a new sub-tree. The new sub-tree is randomly generated.

Step 4: After the predetermined maximum generations are created or a criterion is satisfied, the best individual encountered in the evolution process is selected as the solution.

The reproduction operation is simply the duplication of the original individual probabilistically selected from the population based on fitness. An example illustrating the reproduction process is given in Figure 4.12 of Section 4.3.2.2.

The crossover and mutation operations described above are illustrated in Figure 4.9. For the crossover operation, a crossover point on each parent is randomly selected. On the first parent, node C is selected, and on the second parent, node H is selected. Then the sub-tree rooted at C of the first parent (inside the doted boundary) is replaced with the sub-tree rooted at H from the second parent (inside the doted boundary). The same operation is conducted on the second parent. As a result, two children are produced, as shown in Figure 4.9 (a). For the mutation operation, the node H of the original individual is randomly chosen as the mutation point. Then the sub-tree rooted at H of the original individual is deleted and a new sub-tree grows from the mutation point. The mutated individual is shown in Figure 4.9 (b).

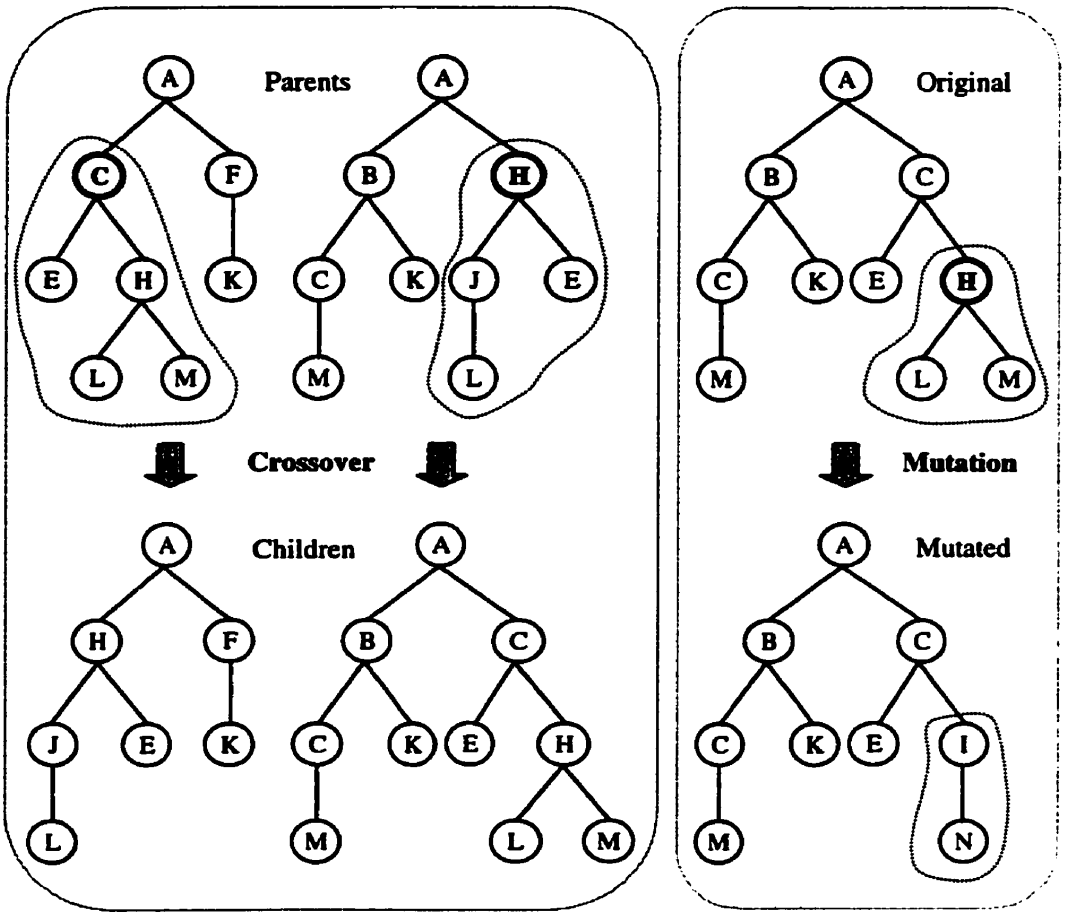


Figure 4.9 Crossover and Mutation Operations in Genetic Programming

In the evolution process, each individual member of a population is evaluated based upon its fitness. The individual member has more chance to survive if it has a better fitness evaluation measure for solving the problem. Usually a fitness function is used to calculate the fitness of individual members. Several formats of fitness functions can be used. If the original function used to evaluate the problem solutions is called raw fitness function $r(x)$, then we have the following fitness function formats:

Standardized Fitness $s(x)$:

$$s(x) = \begin{cases} r(x) & \text{for Min F(x) problems.} \\ r_c / r(x) & \text{for Max F(x) problems. } r_c \text{ is a positive constant.} \end{cases} \qquad (4-2)$$

Adjusted Fitness $a(x)$:

$$a(x) = \frac{1}{1+s(x)} \tag{4-3}$$

Normalized Fitness $n(x)$:

$$n(x) = \frac{a(x)}{\sum_{k=1}^{m} a_k(x)} \tag{4-4}$$

where m is the number of individuals in the population. The normalized fitness reflects the fitness proportion of an individual member in the population. Therefore it can be used as the reference for selecting the corresponding member to take part in evolution operations such as reproduction.

4.3.2.2 Genetic Programming for Alternative Optimization

As described in Section 4.2, product realization process alternatives are described by tree structures. The number of branches and the length of each branch of the alternative trees are different. The method used for alternative optimization should be able to handle the tree structure effectively. The Genetic Programming method is selected in this research because of its advantage in dynamic representation of problem solutions. However, the problem of concurrent design with distributed databases has its own characteristics that require some of the evolution procedures to be modified. Therefore the concept of Genetic Programming plays a more important role in implementing the optimization of product realization process alternatives.

The procedures and related issues using the Genetic Programming method for optimizing product realization process alternatives are discussed in this section.

1. Search Space Representation

Different Internet nodes represent different product development activities. With all possible choices of Internet nodes that are relevant to a given concurrent design problem, alternative combinations of these nodes produce different product realization routes, in other words, different product realization process alternatives. These alternatives compose the search space to be explored for identifying the optimal one. Based on the

definitions of Internet node relations and product realization process alternatives in this research, the search space can be represented using an AND/OR graph. Usually the root node of the tree represents the database for modeling design requirements. The AND/OR graph shown in Figure 4.10 defines twelve product realization process alternatives.

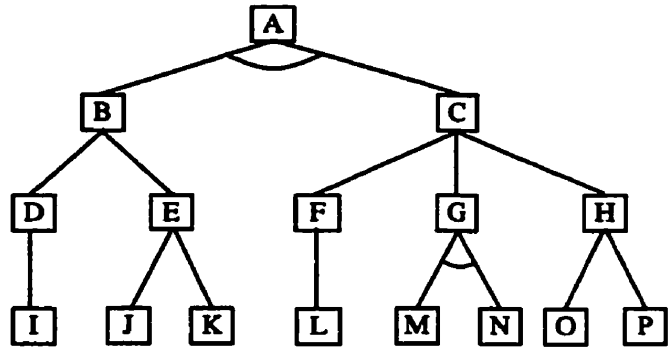


Figure 4.10 The AND/OR Graph Representing a Search Space

2. Generation of Initial Population

The members of the initial population should be randomly generated from the predefined search space. A random alternative is created through the following steps:

Step 1: Identify the root node. Put the root node into an empty list.

Step 2: Pick up a node from the list. This node should be neither a leaf node nor an expanded node. Identify its sub-nodes.

Step 3: If these sub-nodes have an AND relation, put all of them into the list. If these sub-nodes have an OR relation, select one of the nodes randomly and put it into the list.

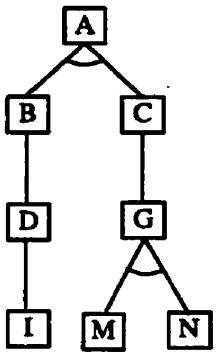
Step 4: Check whether all the nodes in the list are expanded. If no unexpanded node can be found, terminate this process. Otherwise, go to Step 2.

This process is repeated until the required number of individuals is reached. These procedures are similar to the procedures of the exhaustive method for generating all possible alternatives.

Based on the AND/OR graph shown in Figure 4.10, the creation process of a random alternative is illustrated in Figure 4.11.

Step	Nodes in List	Picked Node	All Sub-Nodes	Relation	Selected Sub-Nodes
1	A	A	B,C	AND	B,C
2	A,B,C	B	D,E	OR	D
3	A,B,C,D	C	F,G,H	OR	G
4	A,B,C,D,G	D	I		I
5	A,B,C,D,G,I	G	M,N	AND	M,N
6	A,B,C,D,G,I,M,N				

(a) The Process of Creating a Random Alternative



(b) The Generated Alternative: (A,B,C,D,G,I,M,N)

Figure 4.11 Creation of a Random Alternative

3. Reproduction

The number of times that each individual should be duplicated in the next generation is determined by a probability that is proportional to the individual's fitness. In this research, the normalized fitness *n(x)*, representing the fitness proportion of the selected individual, is used to calculate this number:

$$N_i = \text{int}\left[m \cdot n_i(x)\right], \qquad i = 1, 2, ..., m \qquad (4\text{-}5)$$

where m is the number of individual members in the population. The function int[] converts the real number to its closest integer.

The reproduction operation is illustrated in Figure 4.12. The fitness of each individual is calculated using Equations (4-4) based on the cost value. The number of copies of an individual to be reproduced in the next generation is calculated using Equation (4-5). In this example, two copies of Individual_3 are produced in the next generation, since its fitness is high. Individual_2 died after reproduction because of its very low fitness. The average cost of the population is improved after reproduction.

No.	Population Before Reproduction	Cost	Fitness	Copies	Population After Reproduction	Cost
1	Individual_1	300	0.18	1	Individual_1	300
2	Individual_2	600	0.09	0	Individual_3	100
3	Individual_3	100	0.54	2	Individual_3	100
4	Individual_4	280	0.19	1	Individual_4	280
Average Cost		320				195

Individual_1 (A,B,C,E,G,K,M,N)
Individual_2 (A,B,C,E,G,J,M,N)
Individual_3 (A,B,C,D,H,I,O)
Individual_4 (A,B,C,E,F,J,L)

Figure 4.12 Reproduction of Alternatives

4. Crossover

Crossover is the primary operation for producing new individuals. For concurrent design problems considered in this research, the alternatives, represented by individual members of a population, must follow a predefined syntax. The syntax for the product realization process alternatives refers to the relations defined among the Internet nodes. Specifically, the children produced as the result of crossover operations have to be valid product realization process alternatives. In other words, the syntax of the individuals should be maintained intact after the crossover operations. Therefore the crossover point, the location at which the crossover operation is conducted, must satisfy the following conditions:

(a) The node at the selected location should not be a root node or a leaf node.

(b) The node at the selected location must have OR relation sub-nodes.

(c) The node at the selected location can be found in the other alternative selected for crossover.

Based on the above discussions, crossover operations can be performed when the two selected individuals (alternatives) contain the same Internet node, no matter where

the node is located in each alternative. The procedure for crossover operations is as follows:

(1) The number of the crossover operations, N_c, is calculated by:

$$N_c = \text{int}[0.5(m\text{-}1)P_{c1}] \qquad (4-6)$$

where P_{c1} is a random number between 0 and 1.

(2) The two parent individuals, i.e., alternatives, are chosen randomly from the current population. If the two selected alternatives are marked as A_1 and A_2 respectively, the selection of location on each individual for crossover operation is conducted through the following procedure.

First, if the location of root node (the first node in the node list) is defined as 1, the location of crossover point on alternative A_1 is calculated using the following equation:

$$L_c = \text{int}[(n\text{-}1)P_{c2} + 1] \qquad (4-7)$$

where n is the number of nodes in alternative A_1 and P_{c2} is a random number between 0 and 1. If the node at location L_c does not meet the requirements for crossover, the location is moved one step forward or backward to a new location. The direction of movement is determined randomly. If the node at the new location still cannot satisfy the requirements, the location is continuously moved in the determined direction until a location that meets the conditions for crossover is found. If the location has reached the top (or bottom) of the node list and no valid location is found, then it is relocated to the bottom (or top) of the node list to continue this process.

(3) For each of the parent alternatives A_1 and A_2, identify the nodes on the sub-tree rooted at the selected node. Together with the selected node, these nodes are deleted from their original node list and replaced with the nodes in the sub-tree from the other alternative. Actually, the branch on one alternative tree is

replaced with a branch from the other tree. Then the two child alternatives are produced.

The process of crossover is illustrated in Figure 4.13. The two selected alternatives are marked as Ap_1 and Ap_2 respectively. For alternative Ap_1, if the initial location for crossover is calculated as 4 using Equation (4-7), the node at this location is D. Since node D has only one sub-node (refer to Figure 4.10) and it cannot be found in alternative Ap_2, the location of node D is not valid as a crossover point. Then the location is moved one step forward or backward to a new location of the node list of Ap_1. In this case the next position can be either 3 or 5. The direction of this movement is determined randomly. Supposing the backward direction is selected, the next location is then 3. The node at this location is C, and node C meets all the conditions of a crossover point. So crossover points are determined on both alternatives Ap_1 and Ap_2.

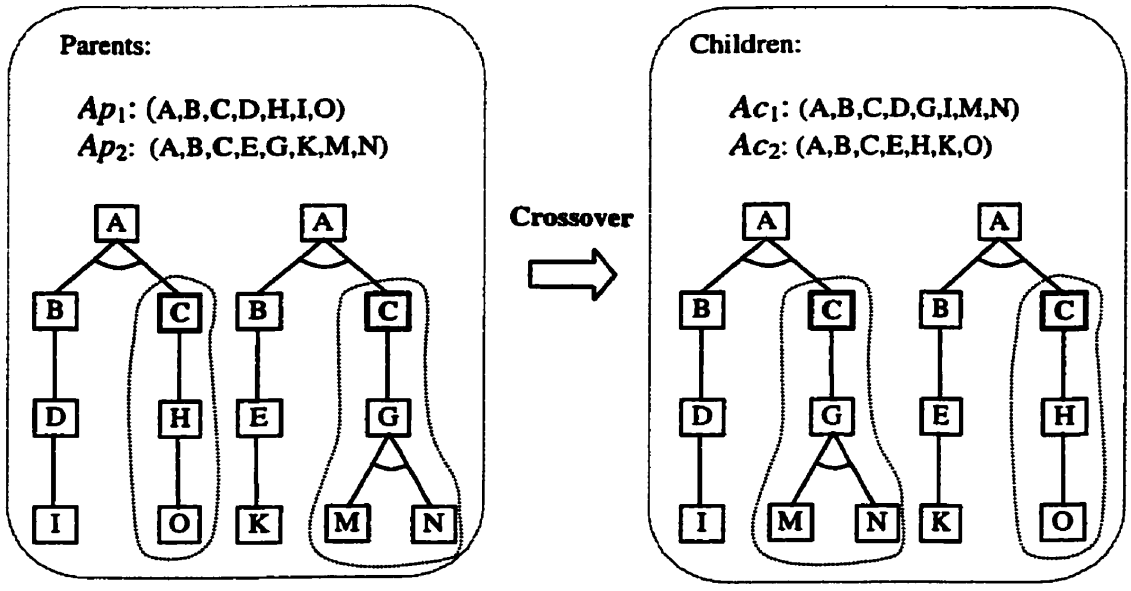


Figure 4.13 Crossover Operation to Alternatives

The nodes on the sub-tree rooted at C on alternative Ap_1 are identified as (C,H,O). These nodes are deleted from the node list of Ap_1 and replaced with the nodes, identified as (C,G,M,N), on the sub-tree rooted at C of alternative Ap_2. The same operation is

conducted for alternative Ap_2. Then the two child alternatives Ac_1 and Ac_2 are produced as shown in Figure 4.13.

5. Mutation

Mutation is another operation for producing new individuals. Each newly produced individual must be correct in the syntax defined for representing the product realization process alternatives. Theoretically, a mutation operation can be conducted at any location on an alternative. To ensure that the sub-tree rooted at the selected node is replaced with a different sub-tree in mutation operation, the selected node must be an OR node. The mutation operation in this research is conducted by the following procedure:

(1) Calculate the number of mutation operations N_{mt} using:

$$N_{mt} = \text{int}[m \cdot P_{mt}] \qquad (4-8)$$

where P_{mt} is a predetermined probability value between 0 and 1.

(2) Pick up an individual randomly from the current population.

(3) Select the location for mutation randomly using the following equation:

$$L_m = \text{int}[(n-1) \cdot P_r + 1] \qquad (4-9)$$

where P_r is a random number between 0 and 1, and n is the number of nodes in the selected alternative. If the node at location L_{mt} is not an OR node, the location is moved one step forward or backward to a new location on the node list. The direction is determined randomly. If the node at the new location is still not an OR node, the location is continuously moved until a valid location for mutation operation is found. The location is valid as a mutation point if the node at this location is an OR node. If the location has reached the top (or the bottom) of the node list of the selected alternative and no valid location is found, then it is relocated to the bottom (or the top) of the node list to continue this process.

(4) Identify all the nodes on the sub-tree rooted at the selected node and delete all these nodes from the node list of the selected alternative.

(5) Select a node randomly from the nodes that have an OR relation with the node at the selected location. A new sub-tree grows with the new node as its root node. The method introduced in Generation of Initial Population for creating a random alternative is used here to produce a random sub-tree. Then the nodes on this new sub-tree are put into the node list of the selected alternative. Now this selected alternative has been mutated.

This process of mutation operation is illustrated in Figure 4.14. If the location of the mutation point of the original alternative is calculated as 7 using Equation (4-9), the corresponding node at this location is M. From Figure 4.10, we can see that M is not an OR node; therefore, this location is not a valid mutation point. To find a valid location, the currently selected location is moved one step forward or backward on the node list of the alternative. The direction of movement is determined randomly. If the direction is determined to be backward in this case, the next location is 6. The node at location 6 is I, which is not an OR node either. So the location is continuously moved backward along the node list to location 5. This location is valid since the corresponding node G is an OR node. The nodes on the sub-tree rooted at G, identified as (G,M,N), are removed from the node list. From the two OR nodes of node G (refer to Figure 4.10), node F is randomly

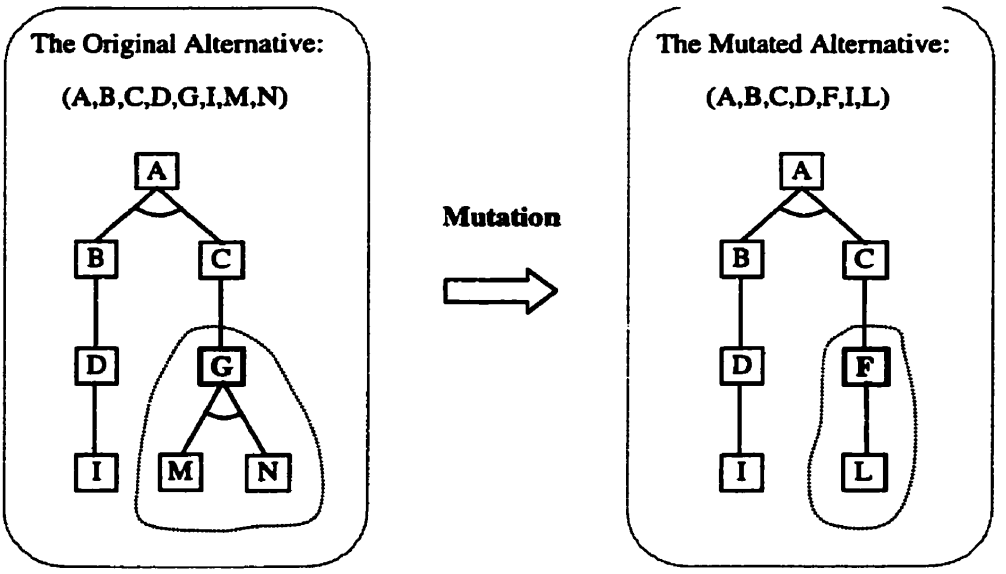


Figure 4.14 Mutation Operation to the Selected Alternative

selected as the new node to replace node G. Using node F as the root node, the new sub-tree (F,L) is generated through the method introduced in Generation of Initial Population. Then the nodes on this newly generated sub-tree are put into the node list. The mutated alternative is shown in Figure 4.14.

6. Solution

For each generation, the best individual (alternative) in the current population is compared with the best individual produced in the previous generations. The better one is selected as the best-so-far alternative. After the predefined number of generations is reached, the evolution process is terminated and the best-so-far individual (alternative) is the solution.

For each alternative produced in the evolution process, parameter optimization is conducted for identifying the optimal parameter values of this alternative. The fitness of the alternative should be calculated with the identified optimal parameter values.

4.4 Identification of Optimal Design Parameter Values

The identification of the optimal product realization process alternative is conducted on the basis that the design parameter values have been optimized in terms of the performance of the design in down-stream product development phases. In other words, for a product realization process alternative, the design parameter optimization must be conducted first, so that the alternative can be evaluated or compared with other alternatives. In this research a population-based optimization method, Particle Swarm Optimization (PSO) [Kennedy and Eberhart 1995, Shi and Eberhart 1998], is adopted for design parameter optimization.

4.4.1 Introduction to Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization method proposed by James Kennedy and Russell Eberhart [Kennedy and Eberhart 1995]. This method simulates social behavior of organisms, such as bird-flocking and fish-schooling.

The idea is that when a bird in a flock tries to find food, it uses not only its own knowledge and experience but also its neighbors' (other birds') experiences.

In PSO, particles fly around in the search space towards the destination (the best position). During flying, each particle adjusts its flying direction and speed according to both its own flying experience and its companions' flying experiences.

If the position of i-th particle is represented as

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{id}), \qquad i = 1, 2, \ldots, N \tag{4-10}$$

where N is the number of particles in the space and d is the dimension of the space. The best previous position of X_i is recorded and represented as

$$P_i = (p_{i1}, p_{i2}, \ldots, p_{id}), \qquad i = 1, 2, \ldots, N \tag{4-11}$$

X_g and P_g are used to represent the best particle (the one with best position) and the best previous position of the best particle respectively. Then we have:

$$X_g = (x_{g1}, x_{g2}, \ldots, x_{gd}) \tag{4-12}$$

$$P_g = (p_{g1}, p_{g2}, \ldots, p_{gd}) \tag{4-13}$$

During flying, the position change (i.e. velocity) for X_i is represented as

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{id}) \tag{4-14}$$

The next position of X_i is then:

$$X_i' = X_i + V_i \tag{4-15}$$

The velocity V_i is calculated by:

$$V_i = wV_{ip} + c_1P_{r1}(P_i - X_i) + c_2P_{r2}(P_g - X_i) \tag{4-16}$$

where V_{ip} is the previous velocity and w is a weighting number. A greater w value results in strong global search ability and a smaller w value leads to a more local search. Coefficients c_1 and c_2 are positive constants. P_{r1} and P_{r2} are two random numbers between 0 and 1. The Equation (4-16) shows that the new velocity of a particle is determined according to its previous velocity, the distance of its current position from its

own best position, and the distance of its current position from the group's best experience (position). Then the particle flies to a new position calculated by Equation (4-15).

PSO is conducted through the following steps:

Step 1: Generate N particles with random positions and velocities in the search space.

Step 2: Evaluate each particle with a pre-defined fitness function that is related to the problem to be solved so X_g and P_g can be identified.

Step 3: Calculate velocity for each particle using Equation (4-16) and the new positions of the particles are determined using Equation (4-15).

Step 4: Repeat Step 2 and Step 3 until pre-determined termination criteria, the maximum position number or the minimum variation of the objective function value, is reached.

According to Shi and Eberhart [1998] and Kennedy and Eberhart [1995], PSO has the following advantages:

1. The concept of PSO is simple and the paradigms of PSO can be implemented in a few lines of computer codes.
2. The methodology of PSO contains evolutionary concepts but needs no coding of problem solutions as does Genetic Algorithm.
3. PSO is computationally inexpensive in terms of both memory requirements and speed because it requires only primitive mathematical operators.

4.4.2 PSO in Design Parameter Optimization

Since this research is concerned with integrating the distributed databases and knowledge bases of different product development processes for concurrent design, the data dependency relation maintenance mechanism is used to propagate data changes in the optimization process.

4.4.2.1 Formulation of Parameter Optimization Problems

Usually an optimization problem can be formulated by an objective function and a collection of constraints. The parameter optimization problems is formulated as:

$$\begin{aligned} &\text{Min} \quad F(\vec{X}) \\ &\text{subject to} \quad : \quad h_i(\vec{X}) \le 0, \quad i = 1, 2, ..., n_h \\ &\qquad\qquad\quad g_j(\vec{X}) = 0, \quad j = 1, 2, ..., n_g \end{aligned} \tag{4-17}$$

where $F(\vec{X})$ is the objective function and $\vec{X}$ represents the design parameters that are usually the attributes of instance features preserved at different Internet nodes. $h_i(\vec{X})$ and $g_j(\vec{X})$ are two types of constraint functions that define the conditions and requirements to the problem to be solved. The numbers of the two types of constraints, $h_i(\vec{X})$ and $g_j(\vec{X})$, are noted as n_h and n_g respectively.

The objective function $F(\vec{X})$ is used to evaluate the performance of the design in down-stream product development processes. It is used directly as the fitness of the particle that is under evaluation. The objective function $F(\vec{X})$ can be described by an equation or by a piece of computer program.

This constrained optimization problem is converted into a non-constrained optimization problem by adding a penalty factor to the objective function. Then a pseudo-objective function in the following form is created:

$$\Phi(\vec{X}) = F(\vec{X}) + W \bullet p(\vec{X}) \tag{4-18}$$

where $p(\vec{X})$ is the penalty function and W is a multiplier constant that determines the magnitude of the penalty. The penalty function takes the following form in this research:

$$p(\vec{X}) = \sum_{i=1}^{n_h} \left[g_i(\vec{X})\right]^2 + \sum_{j=1}^{n_g} \left[h_j(\vec{X}) + \left|h_j(\vec{X})\right|\right]^2 \tag{4-19}$$

With Equations (4-18) and (4-19), violations of constraints will result in a penalty to the original objective function. In other words, if constraints are violated, the fitness of the particle in current position will be low.

4.4.2.2 Issues of Parameter Optimization with PSO

Since the parameter optimization problems involve distributed databases that are represented by instance features, the functions that handle distributed databases, such as informing the changes of virtual attribute values, obtaining virtual attribute values, and propagating data changes automatically, should be accommodated into the optimization process. During optimization, whenever new values are assigned to the design parameters, the calculation for data dependency relation maintenance should be conducted in order to determine the effects of these values on the performance of the design in down-stream product development phases. Since the distributed computing involves communications among related Internet nodes, parameter optimization with geographically distributed databases requires much longer time compared with an optimization with centralized databases.

When PSO is used in design parameter optimization, the basic procedures are the same as those introduced in Section 4.4.1. However the following issues must be addressed:

1. Representation of Design Parameters

Design parameters are represented by attributes of instance features preserved at different locations. In PSO, a group of design parameters is represented as a particle. The different sets of values of the design parameters are represented as the different positions of the particles flying in the search space. Therefore the objective of the problem is to find the best position, i.e., the destination, of the particles.

2. Evaluation of Particles (Positions)

Parameter optimization is conducted, not only to identify the optimal design parameter values, but also to bring the product databases of the selected alternative to an optimal state in terms of concurrent design. To be consistent with the alternative optimization, the evaluation function defined in alternative optimization is used as the original objective function in design parameter optimization.

3. Propagation of Design Parameter Values

Each position of a particle in PSO represents one set of values of the design parameters. When a particle moves to a new position, a new set of values of the parameters is obtained. In order to evaluate these values, the corresponding attributes, no matter where they are located, are updated with the new values by sending messages to the Internet nodes where these attributes are preserved. Then the attribute relation maintenance mechanism is activated automatically to propagate the changes of these attribute values to all related databases. To evaluate the performance of this set of values of the design parameters, fitness is then calculated. If the value of a virtual attribute is required to calculate fitness, a message is sent and the required value is returned through the Internet communication module.

In design parameter optimization using PSO, the values of the constants used in calculation of particle velocity are selected as: $w = 0.8$, $c_1 = c_2 = 2$ according to Shi and Eberhart [1998].

4.4.2.3 The Parameter Optimization Interface

An interface for defining design parameters, constraints and the objective functions has been developed. This interface is named as **Parameter Optimization Window**. The number of particles and the maximum number of positions are also defined in this window. A snapshot of the **Parameter Optimization Window** is shown in Figure 4.15. The configuration of the window is given in Figure 4.16.

The alternative text view is a read-only text view used to display the selected alternative. The three list views are the places for defining design parameters, constraints and objective functions respectively. Commands **Add** and **Remove** are implemented for each of the list views. The two text views in the bottom of the window are used to input the number of particles and the maximum number of positions. All the information displayed in the window can be cleared by clicking on the **Clear** button. After entering all the required information, the optimization process can be started by clicking on the **Start** button.

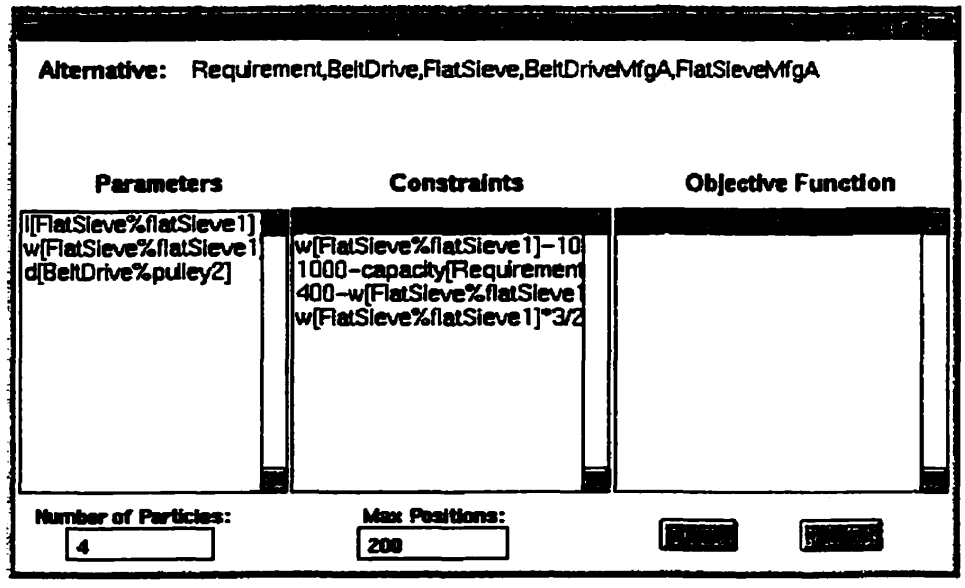


Figure 4.15 A Snapshot of the Parameter Optimization Window

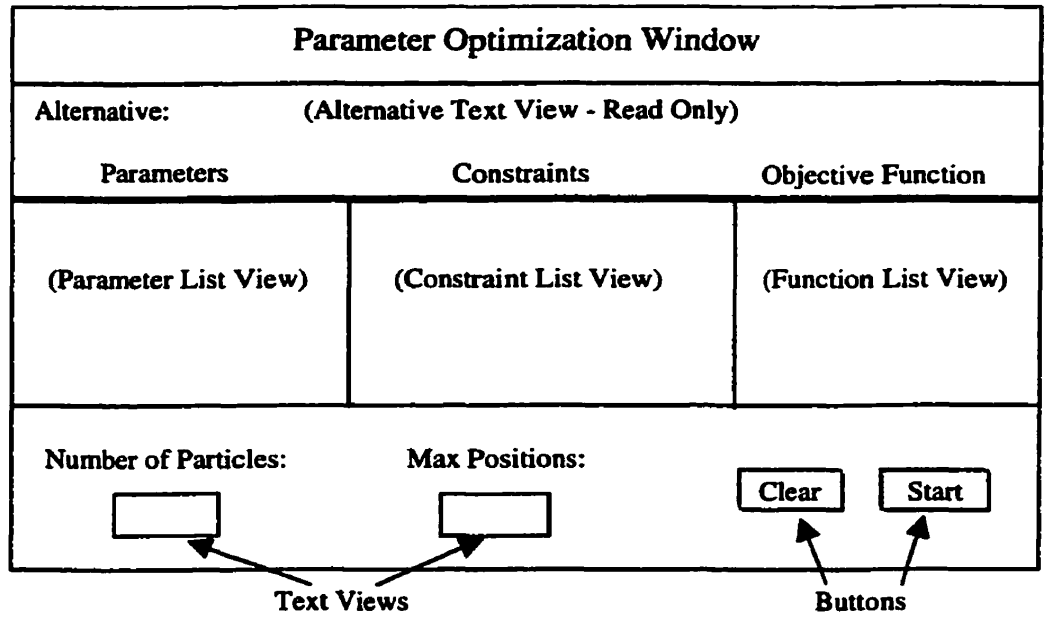


Figure 4.16 Configuration of the Parameter Optimization Window

4.5 Summary

The distributed database and knowledge base modeling system has provided an effective technique for the implementation of concurrent product design with distributed

information resources. Since many resources are available through Internet connections, more alternatives for product realization have brought more chances to reduce product development costs and lead-time.

To identify the optimal alternative for product development with concurrent design technology, two levels of optimization are employed in this research. At the alternative level, two methods are introduced. The exhaustive method is used when the number of alternatives is small. The optimal alternative is selected by comparing all feasible alternatives. When the number of alternatives is large, Genetic Programming method is used to identify the optimal alternative more efficiently. The GP method has the advantage of dynamic representation of the solutions, and therefore is suitable for optimization of the product realization process alternatives. Based on the definition of the product realization process alternatives, techniques used for generating random alternatives, selecting locations for crossover and mutation operations, and search space representation are introduced.

The alternative optimization is conducted based on the results of design parameter optimization. The challenge of design parameter optimization in this research is that the design parameters and the related databases are distributed at different locations. To improve the efficiency of parameter optimization, Particle Swarm Optimization is employed because of the simplicity and quality of the algorithm. The issues of using PSO with distributed design parameters and related databases are addressed. These issues include problem formulation, particle evaluation, and the access of remote data during optimization.

The interfaces developed for accessing the concurrent design system are also introduced in this chapter.

CHAPTER 5

SYSTEM IMPLEMENTATION AND APPLICATION EXAMPLES

This chapter discusses issues in implementing the Internet-based concurrent design system which has been developed based on the distributed database and knowledge base modeling approach described in Chapter 3. These issues include the system interfaces, the data structures, message handling, etc. The system has been implemented using VisualWorks 2.5, which provides a robust Smalltalk application development environment. Application examples are also given in this chapter to illustrate the effectiveness of the introduced methods in distributed database and knowledge base modeling and the concurrent design system.

5.1 System Implementation

5.1.1 System Interfaces

The architecture of the Internet-based concurrent design system has been illustrated in Figure 4.1. Based on this architecture, eight browsers have been developed as the interfaces for this concurrent design system. The functions of each browser and the relations among the eight browsers are shown in Figure 5.1.

The eight browsers of the concurrent design system are managed by a launcher called CDS Launcher as shown in Figure 5.2. CDS stands for Concurrent Design System. All the browsers developed in this system can be activated through this launcher. As shown in Figure 5.3, the browsers are organized into different groups and a drop-down menu is implemented to create the browsers of each group. These browsers can be created by simply clicking on the corresponding menu items.

As described in Section 4.1, the Internet-based concurrent design system is composed of three modules: the concurrent design module, the distributed database and knowledge base modeling module, and the Internet communication module. The Internet communication module is accessed by the Internet Node Definition Browser and the

Node Connection Browser. This module handles the functions related to Internet connections and communications. The Internet node names, addresses, and port numbers defined in the **Internet Node Definition Browser** are necessary information required for data transferring during product development processes. The functions related to data transferring, such as node connections and message handling, are realized through the

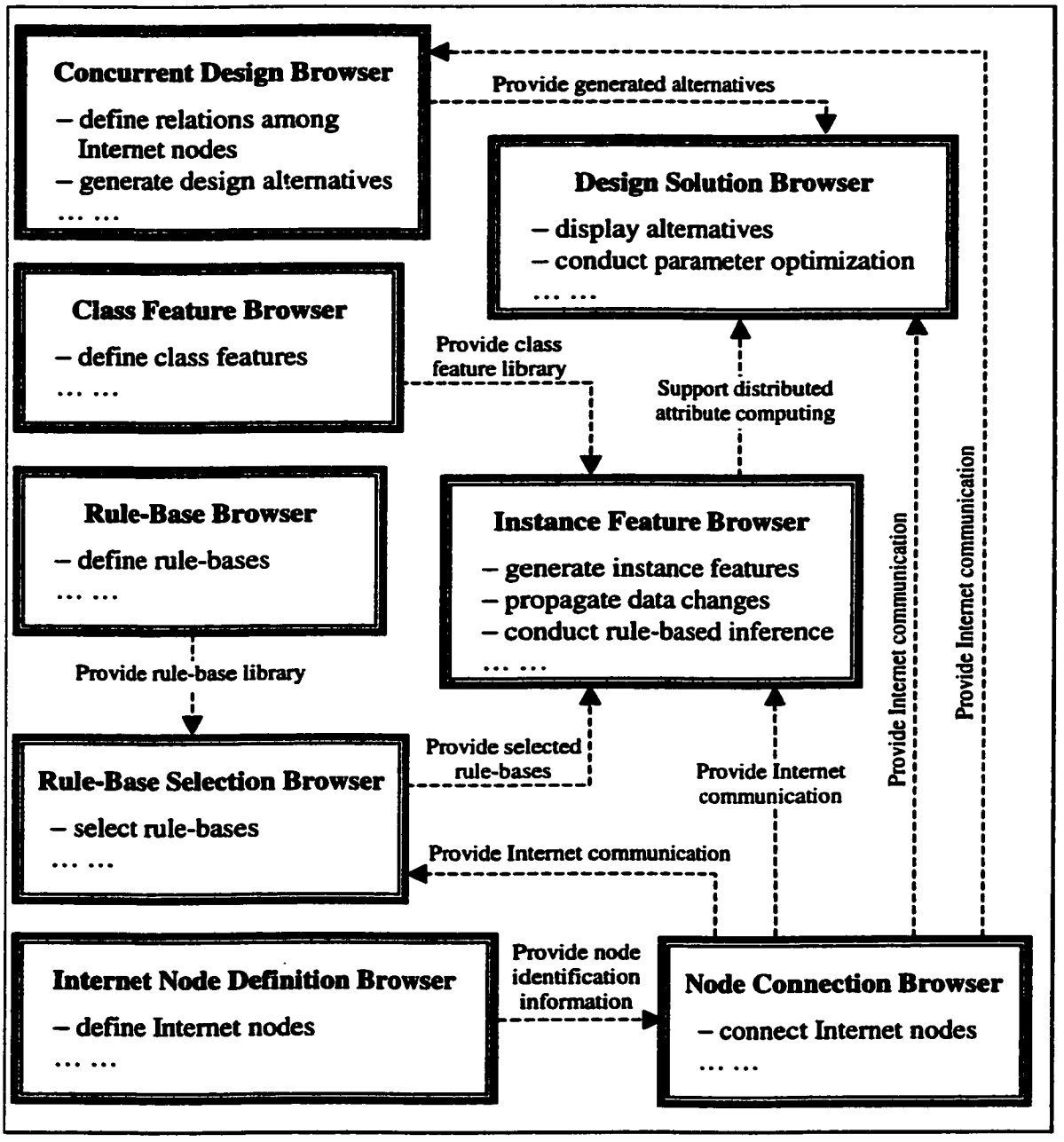


Figure 5.1 The Concurrent Design System Represented by Eight Browsers

Node Connection Browser. This module provides services to both the concurrent design module and the distributed database and knowledge base modeling module.

Figure 5.2 A Snapshot of the CDS Launcher

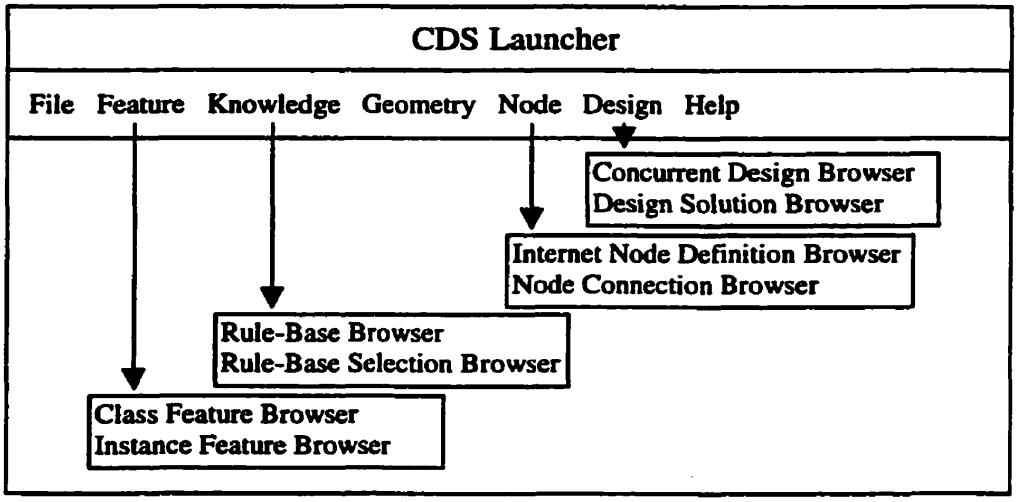


Figure 5.3 Partial Drop-Down Menus of the CDS Launcher

The distributed database and knowledge base modeling module handles the functions such as creating databases and knowledge bases for product development, defining relations among the true data and the virtual data, propagating attribute value changes, and conducting rule-based inference for automating product modeling. This module is accessed by the **Class Feature Browser**, the **Instance Feature Browser**, the **Rule-Base Browser**, and the **Rule-Base Selection Browser**. The **Class Feature Browser** and the **Rule-Base Browser** are used for building the database and knowledge base libraries for product development. Through the **Instance Feature Browser** and the

Rule-Base Selection Browser, the information defined in the libraries is used to produce the product development databases. Functions related to product database modeling, such as generating instance features, propagating changed attribute values, selecting virtual rule-bases, and conducting distributed inference, have been implemented as functions of the **Instance Feature Browser** and the **Rule-Base Selection Browser**.

The concurrent design module provides the functions for conducting product concurrent design, such as modeling product realization process alternatives and identifying the optimal design parameter values and the optimal product realization process. This module is accessed by the **Concurrent Design Browser** and the **Design Solution Browser**. The relations among the involved Internet nodes, defined in the **Concurrent Design Browser**, provide the guidance for automatic generation of valid product realization process alternatives. The **Concurrent Design Browser** also handles the optimization of product realization process alternatives. The generated alternatives are displayed in the **Design Solution Browser**. Parameter optimization is conducted using the **Design Solution Browser**. The concurrent design module is supported by both the distributed database and knowledge base modeling module and the Internet communication module.

5.1.2 New Classes Developed for System Implementation

The concurrent design system has been implemented using VisualWorks [Hopkins and Horan 1995]. VisualWorks provides a large library of classes that can be used for application development. For implementing the concurrent design system, many new classes have been developed in this research. Table 5.1 shows the major newly developed classes.

In addition to the new classes, a number of global variables are defined in the system to preserve the knowledge and data. The names of these variables and the data they represent are listed in Table 5.2. Figure 5.4 shows an example of the typical data structure used in the implemented system. Together with the address and the node name, the port number '8236' for node **BeltDrive** is stored in a node definition object. This object is an

Table 5.1 Major Classes Developed for System Implementation

Modules	Class Names	
Concurrent Design Module	ConcurrentDesignBowser	InternetNodeRelation
	DesignSolutionBrowser	Alternative
	EvaluationFunction	DesignSolution
	Particle	
Distributed Database and Knowledge Base Modeling Module*	ClassFeatureBrowser	FeatureClass
	InstanceFeatureBrowser	FeatureInstance
	RuleBrowser	RuleBase
	RuleBaseSelectionBrowser	SelectedRuleBase
Internet Communication Module	NodeDefinitionBrowser	NodeDefinition
	NodeConnectionBrowser	FeatureSocket

* Most classes in this module were developed in [Yadav 1999], however a large number of new methods have been developed in this research.

Table 5.2 Major Global Variables Used in the Implemented System

Global Variable Names	Data	Remarks
NodesAspectDic	Internet node definition descriptions such as addresses and port numbers	
ConnectedNodesDic	All connected node names	
NodeRelationDic	Descriptions of node relations such as AND relations and OR relations	
DesignSolutionDic	Product realization process alternatives generated in the system	
FeatureCategoryDic	Class features defined in the system	*
FeatureInstanceDic	Instance features generated in the system	*
RulesAspectDic	Rule-bases defined in the system	*

* Implemented in [Yadav 1999]

instance of the class NodeDefinition. This object is then put into the node dictionary with the node name #BeltDrive as the key. This node dictionary is then stored in the category dictionary with the category name #ABC-Food-Industry as the key. Finally, the category dictionary is stored into the Smalltalk system dictionary with the global

variable name #NodesAspectDic as the key. When the port number is requested, the system looks for the key #NodesAspectDic in the system dictionary first and then goes all the way down to the node definition object and gets the requested port number.

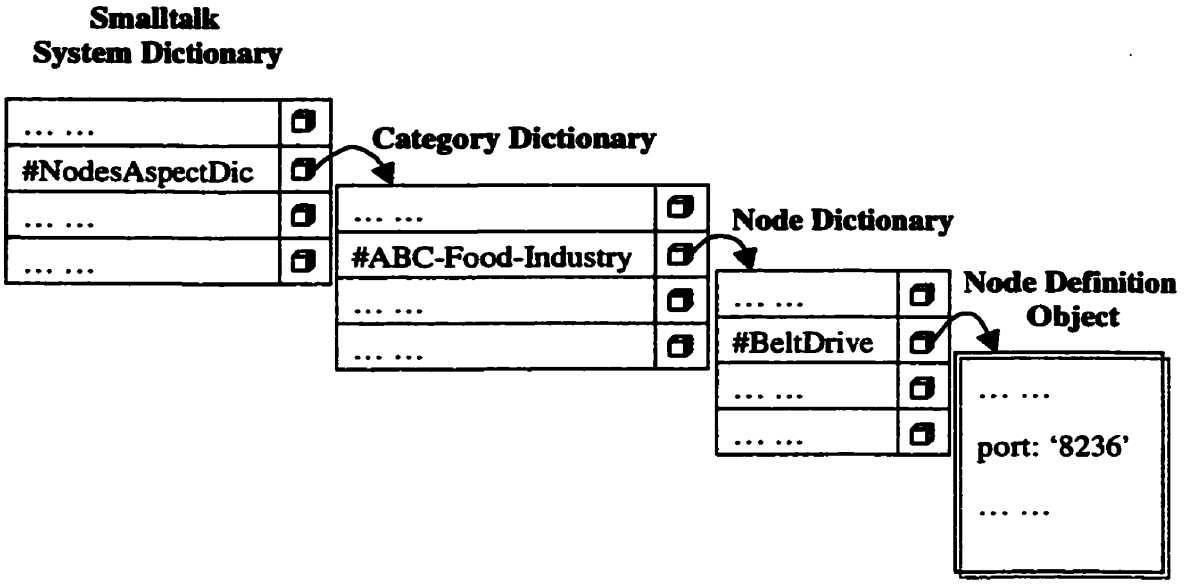


Figure 5.4 A Typical Data Structure Used in System Implementation

5.1.3 Message Handling

In this concurrent design system, information flows from one Internet node to another. The information is described by messages. In order to realize effective communications among involved Internet nodes, the messages are converted to the predicate format:

predicateName(element1,element2,...,elementN)

Usually the name of a predicate states the objective of this message and the elements of the predicate are the data to be transferred or the information required for executing this message. For example, the message updateAttributeValue(gear1,z,30) asks the receiver node to change the attribute z of instance feature gear1 to the new value of 30. Table 5.3 is a list of major predicate messages developed in this system.

A class called **Predicate** is used to convert a *string* message to a standard predicate message and to extract information from the predicate messages.

Table 5.3 Partial Predicate Messages used in the System

Message Names	Purpose of Messages
getAllClassFeatureNames()	Ask a server node for all class feature names.
sendAllClassFeatureNames(*name1,name2, ...*)	Return all the class feature names to a client node.
getAllInstanceFeatureNames()	Ask a server node for all instance feature names.
sendAllInstanceFeatureNames(*name1,name2,...*)	Return all the instance feature names to a client node.
getInstanceFeatureElementNames(*featureName, aspectName*)	Ask a server node for all element names. The corresponding instance feature name and aspect name are specified as the predicate elements.
updateAttributeValue(*featureName,attributeName, newValue*)	Request the receiver node to update an attribute's value. The corresponding instance feature name, attribute name, and the new value are specified as the predicate elements.
... ...	... ...

5.2 Application Examples

In order to illustrate the effectiveness of the distributed database and knowledge base modeling approach and the concurrent design system, application examples in designing a sieving system are given in this section.

Separation of particle materials based on their geometric dimensions is often required in food processing, agricultural engineering, mining and other industries. Such a function can be realized by a mechanical sieving system. A sieving system is usually composed of a power transfer device and a sieving device. When the sieve is in motion, particles with smaller size than the size of the sieve holes can pass through the sieve, and thus are separated from larger particles.

The two design requirements in this example are:

- The capacity of the sieving system is 1000 kg/hour.
- The input rotational speed of the system is 1000 rpm.

5.2.1 The Concurrent Design Problem

The sieving system design problem is composed of two tasks: (1) the design of a power transfer mechanism, and (2) the design of a sieving mechanism. Two alternative power transfer mechanisms, a belt drive mechanism and a gear pair mechanism, and two alternative sieving mechanisms, a flat sieve mechanism and a cylinder sieve mechanism, are considered in this example. These mechanisms are modeled at different Internet nodes as shown in Figure 5.5.

The objective of this design is to find the design alternative that has a minimum manufacturing cost, while satisfying the design requirements. To realize this objective, the manufacturing aspects for the design models shown in Figure 5.5 must be considered during the design process. This is a typical concurrent design problem. In order to consider manufacturing aspects, the relevant manufacturing processes related to the alternative mechanisms are also modeled by different Internet nodes. In this example, two feasible manufacturing nodes, either **BeltDriveMfgA** or **BeltDriveMfgB** as shown in Figure 5.6, can be accessed by the design node **BeltDrive**. These two nodes have an OR relation and are modeled as two sub-nodes of **BeltDrive**. The design requirements are modeled in an Internet node called **Requirement**. To formulate this concurrent design problem, the AND/OR graph with all the involved Internet nodes is formed as shown in

Figure 5.6. The node R1 and node R2 in this graph are not real Internet nodes. They are used to model AND/OR relations of the product realization processes. In this research, this type of nodes is called a pseudo node.

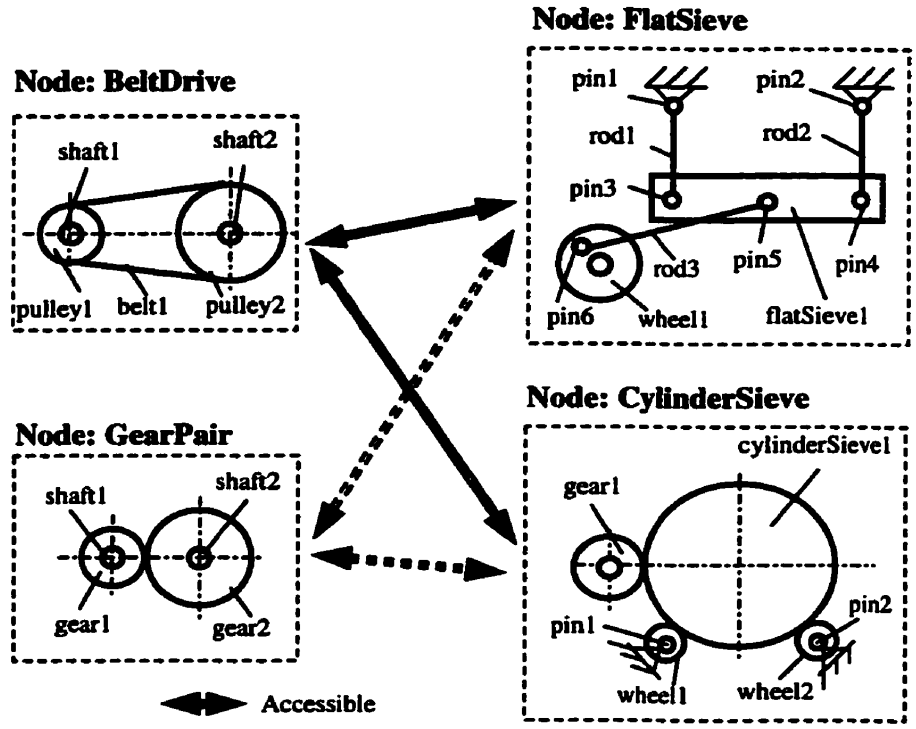


Figure 5.5 Possible Design Alternatives

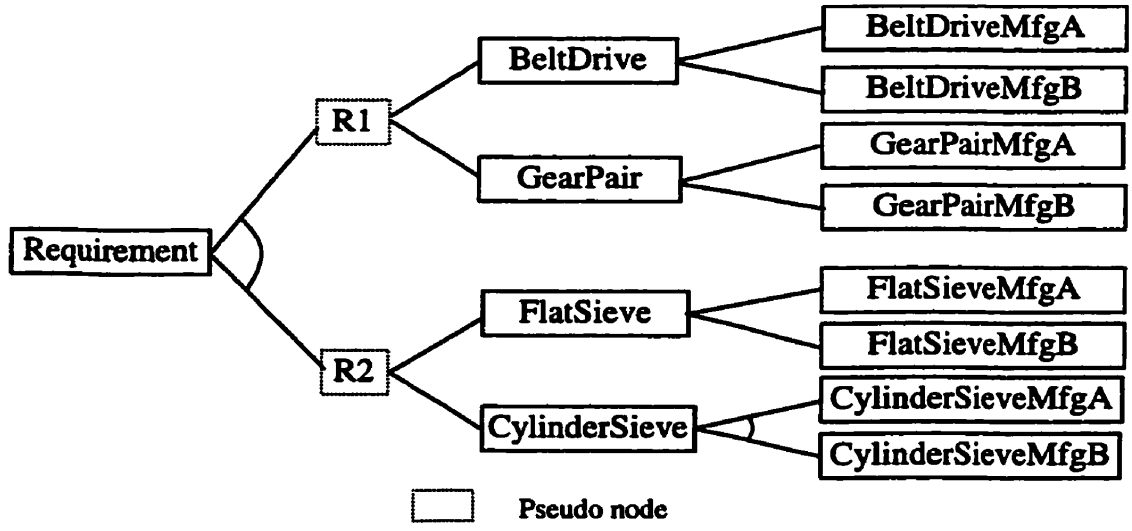


Figure 5.6 The AND/OR Graph Formed with the Involved Internet Nodes

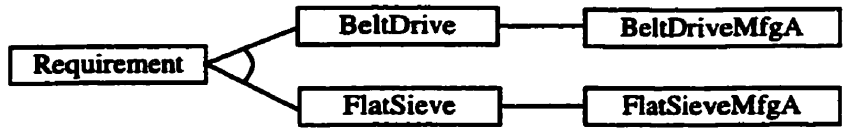
5.2.2 Generation of Instance Features

To implement this example, an IP address and a port number are assigned to each of the involved Internet node. In each Internet node, all required instance features are generated using the corresponding class features. For example, at node **BeltDrive**, the instance feature **beltDrive1** is generated using the class feature **BeltDrive** that is developed for modeling the belt-drive mechanisms. All the element-features of **beltDrive1**, including **pulley1**, **pulley2**, **shaft1** and **shaft2**, are generated using the class features, **Pulley** and **Shaft**, preserved at the local node.

If a required class feature is not available at the local node, a virtual class feature at a remote node can be used to generate the required instance features. For example, at node **CylinderSieve**, instance features **gear1** is required, but there is no corresponding class feature defined at the local node. Therefore, a virtual class feature **Gear** is required to generate the required instance feature. In this case the class feature **Gear** defined at node **GearPair** can be used. After the accessible relation between **GearPair** and **CylinderSieve** is established, the virtual class feature **GearPair%Gear**, which is the class feature **Gear** preserved at node **GearPair**, can be used for generating the instance features **gear1** at node **CylinderSieve**. Partial instance features generated for this example are shown in Figure 5.7 (b).

5.2.3 Rule-Based Reasoning with Virtual Rule-Bases

After all required instance features are generated, attribute values of the generated instance features can be modified manually or through rule-based reasoning. During the database modeling process, if sufficient knowledge bases have been developed, rule-based reasoning can be used to generate or modify the product databases represented by instance features. One of the advantages of the distributed knowledge base modeling approach developed in this research is that the rule-bases preserved in remote nodes can be used for rule-based reasoning at the local node. For example, to design the gear **gear1** in node **CylinderSieve**, two rule bases, **GearMaterial** and **GearProcess**, defined in node **GearPair** for modeling the gears, can be used at node **CylinderSieve**. These two



(a) One Product Realization Process Alternative

Internet Nodes	Instance Features	Attributes
Requirement	customerRequirement1	capacity, cost, inN, ...
BeltDrive	beltDrive1	c, inN, outN, distance, ...
	pulley1	d, n, z, c, mat, ...
	pulley2	d, n, z, c, mat, ...
	shaft1	d, l, n, c, mat, ...
	shaft2	d, l, n, c, mat, ...
FlatSieve	flatSieveMech1	inN, f, capacity, c, ...
	flatSieve1	l, w, f, d, c, ...
	pin1	d, l, c, ...
	pin6	d, l, c, ...
	rod1	d, l, f, c, ...
	rod3	d, l, f, c, ...
	wheel1	d, w, n, c, ...
	...	...
BeltDriveMfgA	beltDriveProcess1	c, ...
	pulley1Process1	d, m, z, c, ...
	p1TurningProcess1	a, t, c, ...
	p1TurningProcess2	a, t, c, ...
	shaft1Process1	d, l, c, ...
	s1TurningProcess1	a, t, c, ...
	s1GrindingProcess1	a, t, c, ...
	... ...	...
FlatSieveMfgA	flatSieveMechProcess1	c, ...
	flatSieveProcess1	l, w, d, c, ...
	fsPunchingProcess1	a, t, d, c, ...
	fsWeldingProcess1	l, c, ...
	pin1Process1	d, l, c, ...
	p1TurningProcess1	a, t, c, ...
	rod1Process1	d, l, c, ...
	r1DrillingProcess1	d, l, c, ...
	r1MillingProcess1	a, t, c, ...
	wheelProcess1	d, w, c, ...
	w1TurningProcess1	a, t, c, ...
	... ...	...

inN: input rotational speed, outN: output rotational speed, d: diameter, n: rotational speed, z: tooth number, l: length, w: width, f: frequency, c: cost, a: area, t: thickness, mat: material.

(b) Partial Instance Features and Attributes at Different Internet Nodes

Figure 5.7 Partial Instance Features Generated at Different Internet Nodes

example rule-bases are shown in Figure 5.8 (a). At node CylinderSieve, the virtual rule

bases GearPair%GearMaterial and GearPair%GearProcess are used for reasoning with local instance feature gear1. The explanation of the rules is given in Figure 5.8 (b). One of the results of rule-based reasoning in this case is that the value of attribute mat[gear1] is modified.

RuleBase: GearMaterial

Rule: MaterialA
IF(gear, ?x) & (<=, m[?x], 2) & (<=, w[?x], 25) THEN (assignAttribute, mat[?x], #AISI1045)

Rule: MaterialB
IF(gear, ?x) & (>=, m[?x], 2.5) & (<=, m[?x], 5) & (>=, w[?x], 25) &(<=, w[?x], 50) THEN (assignAttribute, mat[?x], #AISI2340)

Rule: MaterialC
IF(gear, ?x) & (>=, m[?x], 6) & (>=, w[?x], 25) THEN (assignAttribute, mat[?x], #AISI4140)
...

RuleBase: GearProcess

Rule: GearTypeAndMaterial
IF(gear, ?x) & (gearProcess, ?y) & (elementFeature, ?x, ?y)THEN (assignAttribute, gearType[?y], gearType[?x]) & (assignAttribute, mat[?y], mat[?x])
....

(a) Two Example Rule-Bases

RuleBase: GearMaterial

Rule: MaterialA
IF *there is a gear, and its module number is less than or equal to 2, and its width is less than or equal to 25* **THEN** *set the material of the gear equal to AISI1045.*

Rule: MaterialB
IF *there is a gear, and its module number is greater than or equal to 2.5, and less than or equal to 5, and its width is greater than or equal to 25, and less than or equal to 50* **THEN** *set the material of the gear equal to AISI2340.*

Rule: MaterialC
IF *there is a gear, and its module number is greater than or equal to 6, and its width is greater than or equal to 25* **THEN** *set the material of the gear equal to AISI14140.*
...

RuleBase: GearProcess

Rule: GearTypeAndMaterial
IF *there is a gear, and this gear has a manufacturing process* **THEN** *set the type and material used in the manufacturing process equal to the type and material of the gear.*
...

(b) Rules Explained in Plain English

Figure 5.8 Two Example Rule-Bases for Gear Modeling

5.2.4 Propagation of Changed Attribute Values

When a product realization process alternative, such as (Requirement,BeltDrive, FlatSieve,BeltDriveMfgA,FlatSieveMfgA), has been identified, the Internet node accessibility relations among the involved nodes can be established. For instance, when the accessibility between node **BeltDrive** and node **FlatSieve** is defined, the class and instance features in one node are then virtual class and instance features in another node. The relations among true attributes and virtual attributes are then defined to link the product databases at different Internet nodes. During the design process, the attribute values need to be modified and adjusted. The changes of the attribute values in one node can be propagated to all related attributes including the virtual attributes.

Figure 5.9 shows a partial propagation process started from attribute **d[pulley2]** in node **BeltDrive**. As a result of this propagation process, the values of attribute

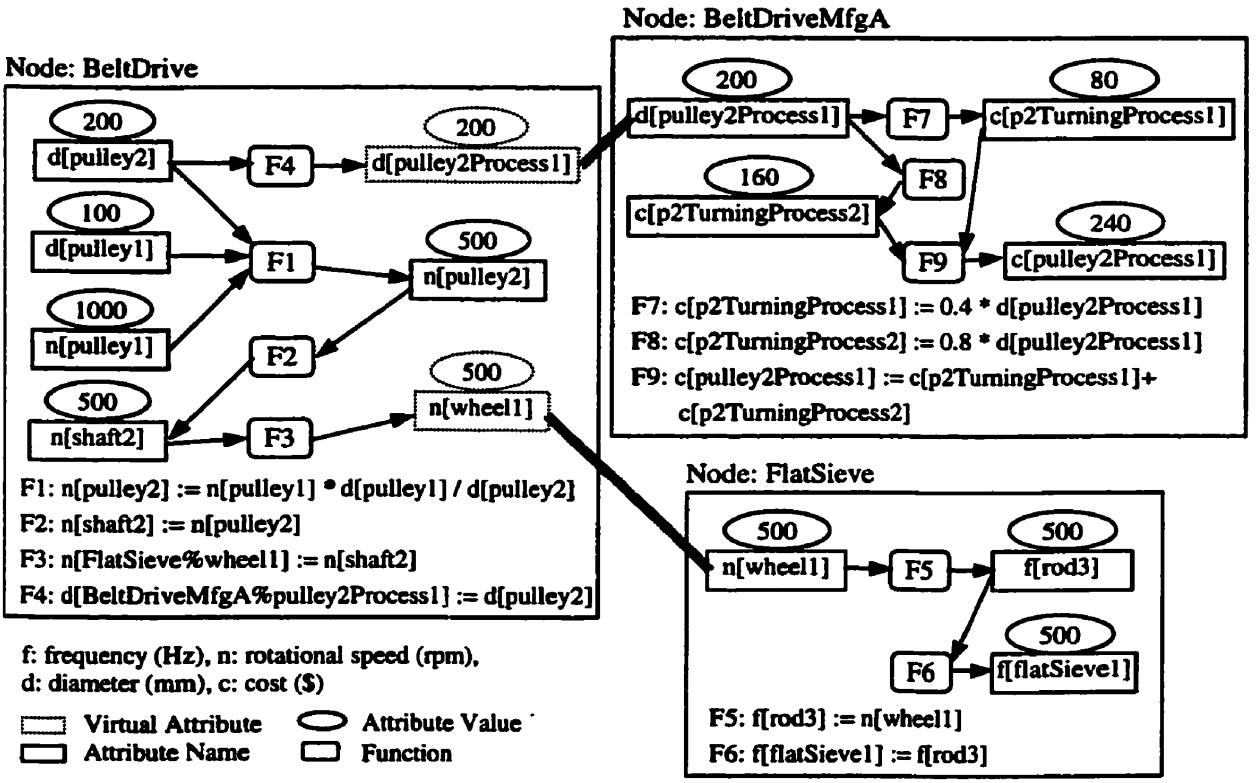


Figure 5.9 Distributed Attribute Relation Network for Modeling the Power Sieving System (F7 and F8 are simplified for modifying the network graph)

f[flatSieve1] in node FlatSieve and the attribute c[pulley2Process1] in node BeltDriveMfgA are updated automatically. Through such automatic attribute change propagation, designers can know the outcomes of the original attribute change as soon as the propagation process finishes. Therefore, the modification to the original attribute value can be evaluated. In the example shown in Figure 5.9, the manufacturing cost of pulley2 is updated automatically if the diameter of pulley2 is changed. So the diameter change of pulley2 can be evaluated by comparing the updated manufacturing cost of pulley2 with the previous cost.

5.2.5 The Optimization of Design Parameter Values Using PSO

The identification of the optimal design parameter values for each alternative is conducted using Particle Swarm Optimization (PSO), as described in Chapter 4. In the example given in this section, the alternative (Requirement,BeltDrive,FlatSieve, BeltDriveMfgA,FlatSieveMfgA), as shown in Figure 5.7 (a), is selected to conduct the parameter optimization.

1. Problem Formulation

The objective of parameter optimization in this example is to determine the values of the selected design parameters so that the sieving system has the minimum manufacturing cost while satisfying the design requirement on system capacity. In this example, three attributes are selected as the design parameters. These attributes are d[BeltDrive%pulley2], l[FlatSieve%flatSieve1], and w[FlatSieve%flatSieve1]. They represent the diameter of pulley2, the length of flatSieve1 and the width of flatSieve1 respectively. The value of attribute n[BeltDrive%pulley1], representing the input rotational speed of the sieving system, is set at 1000 (rpm), based on the design requirements. Then this optimization problem can be formulated as follows:

$$\textbf{Min } F(X) = cost[Requirement\%customerRequirement1]$$

$$X = \{d[BeltDrive\%pulley2],\ l[FlatSieve\%flatSieve1],\ w[FlatSieve\%flatSieve1]\}$$

$$\textbf{Subject to: } l[FlatSieve\%flatSieve1]-1500 \le 0$$

$$w[FlatSieve\%flatSieve1]-1000 \le 0$$

400-w[FlatSieve%flatSieve1] ≤0

*w[FlatSieve%flatSieve1]*3/2- l[FlatSieve%flatSieve1] ≤0*

1000-capacity[Requirement%customerRequirement1] ≤0

In this example, the value of ***F(X)*** is the total manufacturing cost of the belt drive mechanism and the flat sieve mechanism:

cost[customerRequirement1]=c[BeltDriveMfgA%beltDriveProcess1]+
c[FlatSieveMfgA%flatSieveMechProcess1]

The cost of each mechanism is the sum of manufacturing costs of all components of the mechanism:

c[beltDriveProcess1]=c[pulley1Process1]+c[pulley2Process1]+...

c[flatSieveMechProcess1]=c[flatSieveProcess1]+c[pin1Process1]+...

The manufacturing cost of each component is calculated by the cost functions defined in the form of attribute relations of the instance feature representing this component. For instance, the flat sieve is manufactured through two manufacturing processes: the punching process and the welding process. Then the following relations have been defined in node **FlatSieveMfgA**:

F1: *a[fsPunchingProcess1] := l[flatSieveProcess1]*w[flatSieveProcess1]*

F2: *l[fsWeldingProcess1] := (l[flatSieveProcess1]+w[flatSieveProcess1])*2*

F3: *c[fsPunchingProcess1] := a[fsPunchingProcess1]*t[fsPunchingProcess1]**
(d[fsPunchingProcess1]/2)(d[fsPunchingProcess1]/2)*3.14*0.00004*

F4: *c[fsWeldingProcess1] := l[fsWeldingProcess1]*0.025*

F5: *c[flatSieveProcess1] := c[fsPunchingProcess1]+c[fsWeldingProcess1]*

When the length and width of the flat sieve are determined, the punching area *a[fsPunchingProcess1]* and the welding length *l[fsWeldingProcess1]* are calculated first by relation F1 and F2. Then the cost of the punching process *c[fsPunchingProcess1]* and the cost of the welding process *c[fsWeldingProcess1]* are calculated by relation F3 and F4 respectively. The total manufacturing cost of the flat sieve *c[flatSieveProcess1]* is then obtained by relation F5. The costs of other components are calculated in the same way.

2. Design Parameter Optimization

The optimal values of these design parameters are identified by the Particle Swarm Optimization (PSO) algorithm described in Chapter 4. In this example, the dimension of the search space is 3 since three attributes are selected as the design parameters. Therefore, the position of a particle in the search space is represented by the values of the three attributes. The initial positions of particles are randomly assigned. Then the particles fly in the search space towards a target that is the best position of the particles. The flying directions of the particles in the search space are adjusted according to the fitness values of the particles. The fitness values are calculated using Equation (4-4), based, in this example, on the manufacturing cost. The position that the particles land on is the target position, representing the optimal set of values of the design parameters in this concurrent design problem. The sieving system with these attribute values has the minimum manufacturing cost. Since this is an optimization process with distributed parameters, the number of particles affects the optimization efficiency significantly. The greater number of particles defined, the lower the efficiency is. The number of particles chosen in this example is 3. After 200 iterations, the optimal parameter values are obtained as:

$X^* = \{164.785, 990.789, 443.888\}$

$cost[Requirement\%customerRequirement1] = 994.776$

$capacity[Requirement\%customerRequirement1] = 1000.61$

With this set of values, the sieving system has a minimum manufacturing cost and the design requirements are satisfied.

The convergence process is shown in Figure 5.10. The particle fitness is the sum of the original objective function value and the penalty factor defined by Equation (4-19), if constraints are violated. Tests have showed that satisfactory convergence has been achieved.

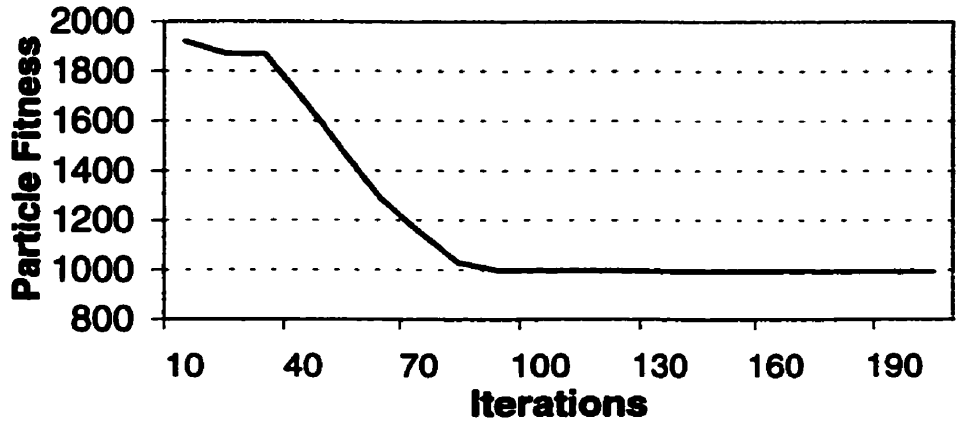


Figure 5.10 The Convergence Process of PSO

5.2.6 The Optimization of Product Realization Process Alternatives Using GP

As shown in Figure 5.6, there are total 12 alternatives for producing the sieving system. The optimal alternative can be identified using either the exhaustive method or the GP method, depending on the number of alternatives. These two methods were introduced in Chapter 4. In this section, the GP method is used to identify the optimal alternative for producing the sieving system.

In this example, manufacturing cost is used as the function to evaluate all alternatives. The costs are calculated in the same procedures described in Section 5.2.5. Therefore the objective is to find a solution alternative that has the minimum manufacturing cost. Based on the graph shown in Figure 5.6, the optimal alternative is identified using the Genetic Programming method, through the following procedures.

1. Generation of the Initial Population

In this example, the number of individuals, representing product realization process alternatives, in the population is 4. The initial population with randomly generated individuals is shown in Figure 5.11. After parameter optimization for each alternative, as described in Section 5.2.5, the manufacturing costs to be used for evaluating the alternatives are obtained. Based on these costs, the fitness of each individual can be

calculated using Equation (4-4). The number of Individuals m is 4, and the adjusted fitness $a(x)$ can be calculated using:

$$a(x) = \frac{1}{1 + cost} \qquad (5-1)$$

The cost and calculated fitness for each alternative are shown in Figure 5.11.

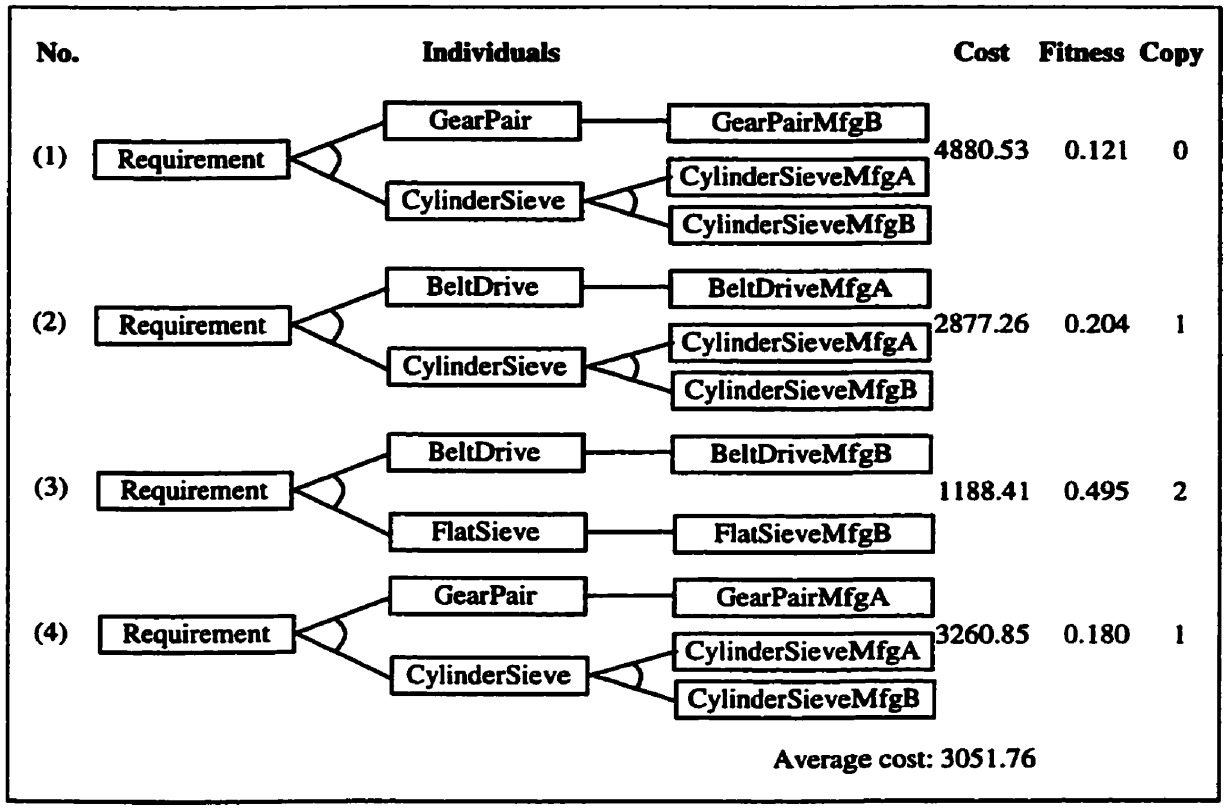


Figure 5.11 The Initial Population

2. Reproduction

The first evolution operation is reproduction. The number of each individual to be copied to the next generation is determined by Equation (4-5). For the initial population, the number that each individual should be duplicated is calculated and shown in Figure 5.11. After reproduction, the individuals in the population are shown in Figure 5.12. Alternative (1) of the initial population died because of its high cost.

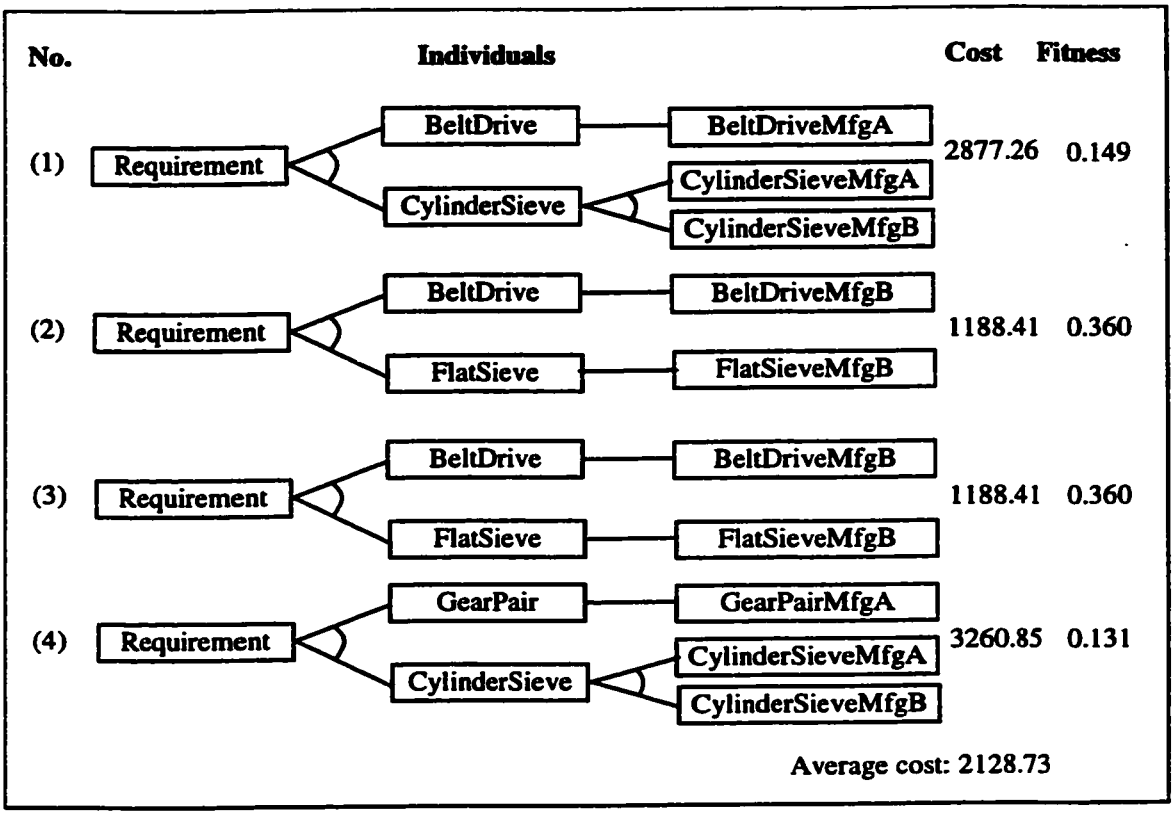


Figure 5.12 The Population after Reproduction

3. Crossover

The number of crossover operation is calculated using Equation (4-6). If the random number P_{c1} is 0.672, then the number of crossover is calculated to be 1.

The selection of the two individuals for crossover is calculated using the following equation:

$$Alternative\ No. = int[(m-1)P_{c2}+1] \qquad (5-2)$$

where P_{c2} is a random number between 0 and 1. If the two random numbers are 0.748 and 0.061, the alternative numbers are calculated as 3 and 1 respectively. So alternative (3) and alternative (1) are selected for crossover.

For alternative (3), the location for crossover is determined by Equation (4-7). In the implemented system, the alternative is described by a list of node names. For example alternative (3) is described as

(Requirement,BeltDrive,FlatSieve,BeltDriveMfgB,FlatSieveMfgB)

For this alternative, the number of nodes is 5. Based on Equation (4-7), if the random number is 0.812, then the location number is calculated to be 4. The node at location 4 is BeltDriveMfgB. Since BeltDriveMfgB is a leaf node, the location of BeltDriveMfgB is not eligible for crossover. To find a new location, the original location, 4 in this case, is moved step by step forward or backward depending on a random number 0 or 1. The random number here is 1, so the location is moved forward one step and the new location is 5. The node at this location FlatSieveMfgB is still a leaf node. Since the location reached the bottom, the next location is 1 where the node Requirement is located. The root node is not eligible for crossover. So the location is moved to 2. The node at location 2 is BeltDrive and it satisfies the conditions for crossover. So the crossover location in both alternatives are determined to be the locations of node BeltDrive in their node lists. Starting from this node, the sub-trees in both alternatives are cut off and switched. In this example, the sub-tree (BeltDrive,BeltDriveMfgB) of alternative (3) and the sub-tree (BeltDrive,BeltDriveMfgA) of alternative (1) are switched. After this operation two new individuals are generated and the new population after the crossover operation is shown in Figure 5.13.

4. Mutation

The number of mutation to be conducted in the current population is determined by Equation (4-8). The mutation probability number is determined as 0.25 in this case. Because the population has 4 individuals, one mutation operation is to be conducted. The alternative selection is conducted the same way as the alternative selection for crossover operations. The random number is 0.976; therefore, alternative (4) is selected. The mutation point on this alternative is selected the same way as the location selection for crossover operations. The selected location is valid as long as the node at the location is an OR node. In other words there are optional choices of nodes to be selected to replace

the node at the selected location on the alternative. In this way, node CylinderSieve is selected.

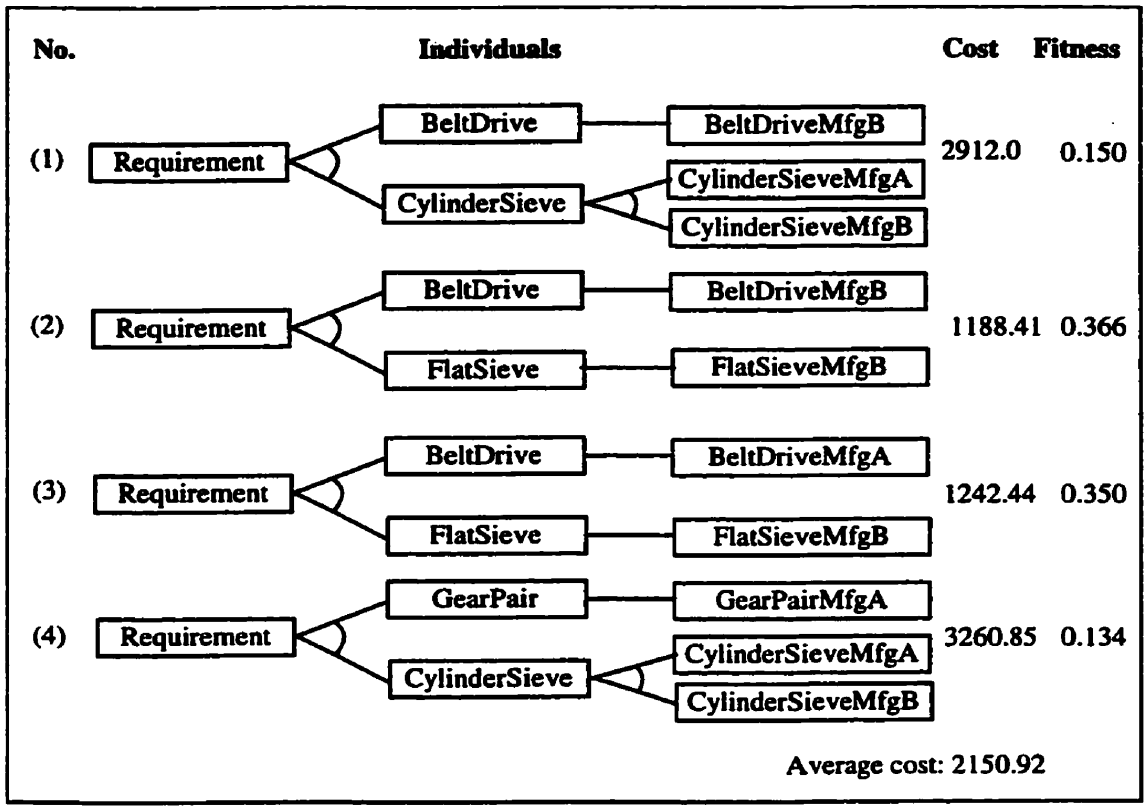


Figure 5.13 The Population after Crossover

To conduct mutation, the sub-tree rooted at the node CylinderSieve in individual (4) shown in Figure 5.13 is cut off and replaced by a new node. This new node is randomly selected from the OR nodes of the selected node. In this case there is only one choice, i.e., node FlatSieve. Starting from this new node, a new sub-tree, (FlatSieve,FlatSieveMfgA), grows so that a new individual is generated. After the mutation operation, the second generation of the population is produced, as shown in Figure 5.14.

Comparing the average cost of all alternatives in the second generation with the average cost in the first generation, the quality of the population has been improved in terms of the manufacturing cost. The above evolution process is continued until the

predetermined generation number is reached. Then the best alternative recorded in the evolution process is the solution. In this example, the alternative (Requirement, BeltDrive,FlatSieve,BeltDriveMfgA,FlatSieveMfgA) is identified to be the optimal alternative for realizing the sieving system, in terms of the minimum manufacturing cost. The manufacturing cost for this alternative is 944.776.

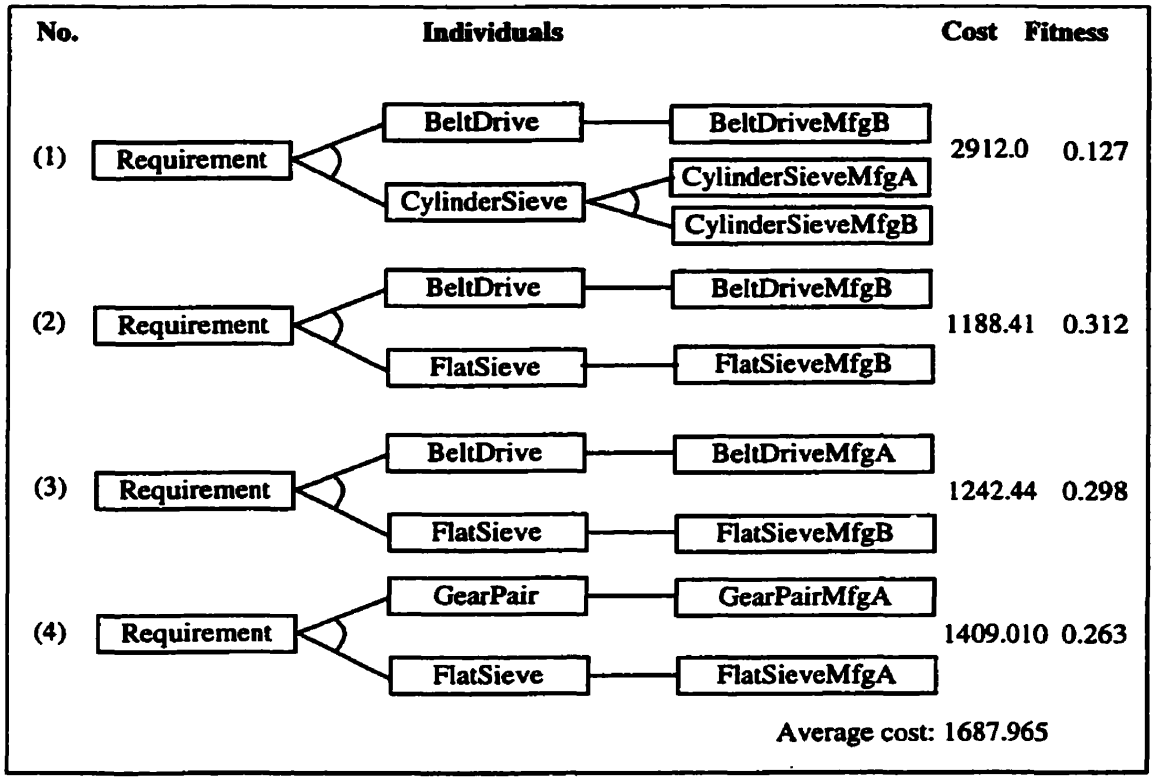


Figure 5.14 The Population after Mutation: The Second Generation

The concurrent design process in this exampleshows that the distributed database and knowledge base modeling system is effective for product development with concurrent design methodology. The identification of optimal alternative and design parameter values can be easily realized using the concurrent design system developed in this research.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this chapter, conclusions drawn from this research are summarized, and related future work is outlined.

6.1 Conclusions

Concurrent design is a design methodology by which the down-stream product development life-cycle aspects are considered concurrently at the design stage. With the rapid development of Internet technology, information resources geographically distributed at different locations are now available for product development at a local location. More alternative processes are available for producing a product, because of a wider choice of information resources from different locations. This research was devoted to the development of a feature-based distributed database and knowledge base modeling approach and an Internet-based concurrent design system. The conclusions drawn from this research are summarized in the following sections.

6.1.1 Distributed Database and Knowledge Base Modeling

(1) The feature-based product development life-cycle databases and knowledge bases distributed at different locations can be integrated dynamically through the Internet.

Conventional product development approaches such as Design for "X" and concurrent design were developed based on centralized computing techniques [Kusiak 1993]. In this research, physically distributed product development life-cycle databases and knowledge bases are integrated through the Internet. These databases and knowledge bases are used for modeling product development life-cycle activities. The distributed product development activities are modeled at different Internet nodes. An Internet node can be added to or removed from the distributed database and knowledge base modeling system by connecting and

disconnecting the node to the system. This function can be used to access the globally available information resources for product development.

(2) *Distributed product development life-cycle databases, modeled by both true and virtual instance features, can be associated by defining relations among these data.*

Traditionally, distributed database management systems are required to manage databases distributed at different locations [Bray 1982]. In this research, the distributed databases are associated directly by defining relations among the true and virtual features. Though the Internet provides the physical connections, the defined relations among the distributed features and attributes integrate the distributed databases into the same environment. The data relations can be modified conveniently during the product development process to identify the optimal design, considering the down-stream product development life-cycle aspects.

(3) *Product development database libraries at remote nodes can be used to generate product modeling databases at the local node to improve product development efficiency.*

Instance features distributed at different locations are generated using corresponding class features as their templates. If the required class feature is not defined at the local location, a virtual class feature can be used to generate true instance features directly at the local site. This function provides support to product data library sharing; therefore, the efficiency of product development is improved.

(4) *Consistency of distributed product databases can be maintained using the distributed data dependency relation maintenance mechanism.*

The consistency of the distributed product databases is maintained using the distributed data dependency relation network. To propagate the data changes to related data including virtual data, a distributed data dependency relation maintenance mechanism has been developed in this research. During the product development process, modifications to product development data are necessary. When part of the data are modified, all the related data at the local site and the

remote sites are updated automatically. During product development, when the design data are changed, manufacturing process descriptions are updated using the relations between design and manufacturing descriptions. The updated manufacturing aspects can be used to evaluate the design. This function is very effective in modeling product databases and product development life-cycle aspects for concurrent design.

(5) *Knowledge bases preserved at remote locations can be used for knowledge-based inference at the local location.*

The knowledge bases distributed at different locations can also be integrated through the Internet. Virtual rule-bases, the rule-bases preserved at remote nodes, can be used for rule-based reasoning at the local node. Virtual rule-bases are also visible at the local node. The selection of virtual rule-bases for reasoning at the local node is accomplished through Internet communications. A virtual rule-base can be selected and removed at the local node using a specially developed browser. This mechanism is effective when no sufficient knowledge bases are provided at the local site.

(6) *The efficiency of product development with distributed databases and knowledge bases can be improved by using the distributed rule-based inference mechanism.*

The process of product modeling using distributed databases can be automated by rule-based inference. Since rule-based reasoning may result in data change in remote nodes, a distributed inference mechanism has been developed in this research. When the data in a remote node are changed as a result of local rule-based reasoning, the rule-based reasoning in that node is automatically activated. This function is effective for creating distributed product descriptions.

6.1.2 Internet-Based Concurrent Design

(1) *The Internet-based concurrent design system developed in this research can model alternative product realization processes.*

Based on the distributed database and knowledge base modeling approach, an Internet-based concurrent design system has been developed for product concurrent design using the distributed product development life-cycle databases. Modeling of product realization process alternatives is essential for conducting product concurrent design using a computer-based concurrent design system. In this research, the different product development life-cycle aspects are modeled at different Internet nodes. The product realization process alternatives are modeled by AND/OR graphs in which the nodes are used to represent different product development activities distributed at different locations. This product realization process alternative modeling approach is effective for generating and evaluating the alternatives to identify the optimal one.

(2) *The optimal design parameter values, considering the manufacturability of the design, can be identified using a global optimization method.*

For a generated product realization process alternative, the optimal design parameter values can be identified using the Particle Swarm Optimization (PSO) method employed in the implemented system. In PSO method, different positions of the particles flying in the search space represent different sets of values of the design parameters. For each set of new values, automatic data change propagation is conducted to update the related data values. Then the updated results can be used to evaluate the current position of the particle. For a concurrent design problem, the design parameter values can be continuously evaluated using the manufacturability measures during the optimization process. The design parameter values identified by PSO are optimal in terms of the manufacturability of the products. In this research, PSO is efficient and reliable in optimizing the design parameters distributed at different locations.

(3) *The optimal product realization process alternative can be identified by two different methods: the exhaustive method and the Genetic Programming (GP) method.*

Among the feasible product realization process alternatives, the optimal alternative can be identified by two different methods: the exhaustive method and the Genetic

Programming method. If the number of alternatives is small, the exhaustive method can be used to generate all feasible alternatives. These alternatives can then be evaluated and compared to identify the optimal one. An algorithm has been developed for generating all feasible alternatives. If the number of alternatives is large, the Genetic Programming method can be used to identify the optimal alternative. Modifications to the GP algorithm have been made for solving concurrent design problems. The modified GP method is effective in alternative optimization.

6.2 Future Work

The distributed database and knowledge base modeling approach and the Internet-based concurrent design system developed in this research are effective for engineering product design with distributed resources. However, this work can be further improved in the following aspects:

(1) Improvement using multi-agent systems

In this research project, the selection of relevant databases and knowledge bases for product development is conducted manually by users. The algorithms introduced for the attribute value change propagation and the distributed inference may be not efficient when a complex project is involved. With the advances of multi-agent systems [Norrie 1999, Shen and Norrie 1999, Shen et al. 1999, Shen et al. 2000, Ulieru et al. 2000], overall system performance can be improved if the functions, such as Internet communications, data dependency relation maintenance, distributed inference, rule-base selections, and alternative optimization, are handled by agents, especially when large amount of information and operations are involved in the design process.

Product concurrent design with distributed databases and knowledge bases is a complex process involving a wide range of technical and social knowledge. Different types of autonomous agents with different knowledge can be used to handle different

aspects of the product development life-cycle. Through coordination and cooperation among the involved agents, the goals of product concurrent design can be achieved.

(2) Product geometric representation

This research focuses on modeling functional design aspects of the products. However the representation of product 2D and 3D geometry is another important aspect in product development. The existing system can be enhanced if a product geometry modeling module is developed. Transformation of the feature-based product geometry data into a format understandable by commercially available CAD software should be a subject for further research.

(3) System interfaces

The interfaces developed in this research are Smalltalk browsers. The input and output information is handled using text views. To improve the interface environment, graphical windows with functions for defining and displaying the Internet node, the product databases and knowledge bases, as well as product realization process alternatives can be introduced. Using web browsers to access the concurrent design system can also improve the interface environment.

(4) Incorporation of web techniques

The concurrent design system developed in this research has the potential to be incorporated into a web-based product development environment. In this environment, the system can be easily accessed through computers connected to Internet; therefore, accessibility to the concurrent design system can be improved.

(5) Improvement in alternative optimization

In this research, the Genetic Programming method has been employed for identifying the optimal product realization process alternative. In the optimization process, the alternatives are dynamically generated by the GP method. For each new alternative, parameter optimization must be conducted to bring the alternative to an optimal state, so that this alternative may be compared with others. Therefore the alternative

optimization is conducted generation by generation with user interference. New approaches should be studied to improve efficiency by combining the two optimization processes without human interference.

REFERENCES

Adamides, E. D., 1995, Coordination of Distributed Production Resources for Responsibility-Based Manufacturing. *Journal of Intelligent Manufacturing*, Vol. 6, No. 6, pp. 415-427.

Adapalli, S. and Addepalli, K., 1997, World Wide Web Integration of Manufacturing Process Simulations. *Concurrency: Practice and Experience*, Vol. 9, No. 11, pp. 1341-1350.

Ahn, S.–H., Roundy, S., Wright, P. K., and Liou, S.–Y., 1999, 'Design Consultant': A Network-Based Concurrent Design Environment. *MED 10, American Society of Mechanical Engineers,* Manufacturing Engineering Division, ASME, November, pp. 563-569.

Alles, D. and Vergottini, G., 1997, Taking a Look at Internet-Based Design in the Year 2001. *Electronic Design*, January, pp. 42-50.

Anderson, D. C. and Crawford, R. H., 1988, Knowledge Management for Preliminary Computer-Aided Mechanical Design. *Organization of Engineering Knowledge for Product Modeling in Computer Integrated Manufacturing*, (ed.), T. Sata, Elsevier, pp. 15-38.

Angeline, P. J., 1994, Genetic Programming: A Current Snapshot. *Proceedings of the Third Annual Conference on Evolutionary Programming*, (eds.), A. Sebald and L. Fogel, World Scientific, River Edge, NJ, pp. 224-232.

Arora, J. S., Elwakeil, O. A., and Chahande, A. I., 1995, Global Optimization Methods for Engineering Applications: A Review. *Structural Optimization*, Vol. 9, pp. 137-159.

Bassiliades, N. and Vlahavas, I., 1997, Processing Production Rules in DEVICE, An Active Knowledge Base System. *Data & Knowledge Engineering*, Vol. 4, pp. 117-155.

Bliznakov, P. I., Shah, J. J., Jeon, D. K., and Urban, S. D., 1995, Design Information System Infrastructure to Support Collaborative Design in a Large Organization. *Proceedings of the 1995 ASME Design Engineering Technical Conferences*, Boston, Vol. 1, pp. 1-8.

Bray, O. H., 1982, *Distributed database management systems*, Lexington Books, Lexington, Mass.

Chen, Y.–M. and Jan, Y.–D., 2000, Enabling Allied Concurrent Engineering through Distributed Engineering Information Management. *Robotics and Computer-Integrated Manufacturing*, Vol. 16, No. 1, pp. 9-27.

Chung, J. C. H., Patel, D. R., and Cook, P. L., 1990, Feature-Based Modeling for Mechanical Design. *Computer & Graphics*, Vol. 14, No. 2, pp. 189-199.

Colton, J. S., 1993, An Intelligent Design for Manufacture System. *Concurrent Engineering: Automation, Tools, and Techniques*, (ed.), A. Kusiak, John Wiley & Sons, Inc.

Court, A. W., 1998, Issues for Integrating Knowledge in New Product Development: Reflections from an Empirical Study. *Knowledge-Based Systems*, Vol. 11, pp. 391-398.

Cutkosky, T. G., Tenenbaum, M. R., and Glicksman, J., 1993, SHARE: A Methodology and Environment for Collaborative Product Development. *Proceedings of IEEE Infrastructure for Collaborative Enterprise*, IEEE, Morgantown, pp. 33-41.

Danesh, M. R. and Jin, Y., 1999, AND: An Agent-Based Decision Network for Concurrent Design and Manufacturing. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Domazet, D. S., Kong, H. P. H., Yan, M. C., Calvin, C. F. Y., and Goh, A., 2000, Infrastructure for Inter-Organization Collaborative Product Development. *Proceedings of the Hawaii International Conference on System Sciences*, January, pp. 159.

Domazet, D. S. and San, L. S., 1997, Active Database Servers for Concurrent Engineering Environments. *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, Melbourne, Australia.

Dong, Z. (ed.), 1994, *Artificial Intelligence in Optimal Design and Manufacturing*, PTR Prentice Hall.

Gardan, Y. and Minich, C., 1993, Feature-Based Models for CAD/CAM and Their Limits. *Computers in Industry*, Vol. 23, pp. 3-13.

Gadh, R. and Sonthi, R., 1998, Geometric Shape Abstractions for Internet-Based Virtual Prototyping. *Computer-Aided Design*, Vol. 30, No. 6, pp. 473-486.

Goldberg, A. and Robson, D., 1983, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Grimes, R., 1997, *Professional DCOM Programming*, Wrox Press.

Gu, P. and Chan, K., 1995, Product Modeling Using STEP. *Computer-Aided Design*, Vol. 27, No. 3, pp. 163-179.

Hahn, H. and Stout, R., 1994, *The Internet: Complete Reference*, McGraw-Hill.

Helander, M. and Nagamachi, M., 1992, *Design for Manufacturability, A Systems Approach to Concurrent Engineering and Ergonomics*, Taylor & Francis Ltd.

Henderson, M. R., 1984, Extraction of Feature Information from Three Dimensional CAD Data. Ph.D. Dissertation, Purdue University.

Higgins, K. B. and Langrana, N. A., 1999, Web-Based, User-Friendly Design and Virtual Fabrication for Layered Manufacturing. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Hopkins, T. and Horan, B., 1995, *Smalltalk: An Introduction to Application Development Using VisualWorks*, Prentice Hall.

Huang, G. Q., Huang, J., and Mak, K. L., 2000, Agent-Based Work-Flow Management in Collaborative Product Development on the Internet. *Computer-Aided Design*, Vol. 32, pp. 133-144.

Huang, G. Q., Lee, S. W., and Mak, K. L., 1999, Web-Based Product and Process Data Modeling in Concurrent "Design for X". *Robotics and Computer-Integrated Manufacturing*, Vol. 15, pp. 53-63.

Hughes, J. G., 1991, *Object-Oriented Databases*, Prentice-Hall.

Huhns, M. N. and Singh, M. P., 1998, *Readings in Agents*, Morgan Kaufmann Publishers.

Hyeon, H. J., Hamid, R. P., and Sullivan, W. G., 1993, Principles of Concurrent Engineering. *Concurrent Engineering*, (eds.), H. R. Parsaei and W. G. Sullivan, Chapman & Hall.

Jiang, P.–Y. and Fukuda, S., 1999, Internet Service and Maintenance for RP-Oriented Tele-Manufacturing. *Concurrent Engineering: Research and Applications*, Vol. 7, No. 3, pp. 179-189.

Judson, J., Dong, Q., and Mascoli, G., 1999, Introducing Knowledge-Based Engineering into an Interconnected Product Development Process. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Kennedy, J. and Eberhart, R., 1995, Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia.

Kim, C.–Y., Kim, N., Kim, Y., Kang, S.–H., and O'Grady, P., 1998, Distributed Concurrent Engineering: Internet-Based Interactive 3-D Dynamic Browsing and Markup of STEP Data. *Concurrent Engineering: Research and Applications*, Vol. 6, No. 1, pp. 53-70.

Kim, J. H., Wang, F.–C., Sequin, C. H., and Wright, P. K., 1999, Design for Machining over the Internet. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Koza, J. R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press.

Kroll, E., Lenz, E., and Wolberg, J. R., 1989, Knowledge-Based Synthesis in Design-for Assembly. *Concurrent Product and Process Design*, (eds.), N. H. Chao and S. C.–Y. Lu, The American Society of Mechanical Engineers.

Kusiak, A. (ed.), 1993, *Concurrent Engineering: Automation, Tools, and Techniques*, John Wiley & Sons.

Lee, E. T. Y., 1985, Some Remarks Concerning B-Splines. *CAGD Journal*, Vol. 2, pp. 145-149.

Lee, J. Y., Kim, H., and Han, S.–B., 1999, Web-Enabled Feature-Based Modeling in a Distributed Design Environment. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Lee, K. H. and Sen, S., 1994, ICOSS: A Two-Layer Object-Based Intelligent Cell Control Architecture. *Computer Integrated Manufacturing Systems*, Vol. 7, No. 2, pp. 100-112.

Magrab, E. B., 1997, *Integrated Product and Process Design and Development – the Product Realization Process*, CRC Press LLC.

Mendel, A., 1999, PDM and the Internet. *Mechanical Engineering*, September.

McCarty, B. and Cassady-Dorion, L., 1999, *Java Distributed Objects*, SAMS.

Mortenson, M. E., 1985, *Geometric Modeling*, John Wiley, New York.

Nagamatsu, M., Sumida, S., and Nagamatsu, A., 1999, A New Approach on Modeling for Product Development. *JSME International Journal*, Series C, Vol. 42, No. 1, pp. 234-

239.

Name, E. V. and Eagelstein, G., 1998, The Wired Engineer: Emerging Technologies and the Designer. *ANTEC '98*, pp. 3052-3055.

Norrie, D. H., 1999, *Multi-Agent Systems*, Lecture Notes, The University of Calgary.

Otte, R., Patrick, P., and Roy, M., 1996, *Understanding CORBA: The Common Object Request Broker Architecture*, Addison-Wesley.

Ozsu, M. T., Dayal, U., and Valduriez, P., 1994, *Distributed Object Management*, Morgan Kaufmann Publishers, San Mateo, California.

Pahng, F., Senin, N., and Wallace, D., 1998, Distribution Modeling and Evaluation of Product Design Problems. *Computer-Aided Design*, Vol. 30, No. 6, pp. 411-423.

Pardalos, P. M., Romeijin, H. E., and Tuy, H., 1999, Recent Developments and Trends in Global Optimization. *Research Report* 99-15, Department of Industrial & System Engineering, University of Florida.

Park, H. G. and Baik, J. M., 1999, Enhancing Manufacturing Product Development through Learning Agent System over Internet. *Computer and Industry Engineering*, Vol. 37, No. 1, pp. 117-120.

Parsaei, H. R. and Sullivan, W. G., 1993, *Concurrent Engineering*, Chapman & Hall.

Pennel, J. P. and Winner, R. I., and Slusarczuk, M. M. G., 1989, Concurrent Engineering: An Overview for Autotestcon. *AUTOTESTCON Proceedings '89: The System Readiness Technology Conference*, Philadelphia, PA, pp. 88-99.

Penoyer, J. A., Burnett, G., Fawcett, D. J., and Liou, S.–Y., 2000, Knowledge Based Product Life Cycle Systems: Principles of Integration of KBE and C3P. *Computer-Aided Design*, Vol. 32, pp. 311-319.

Prasad, B., 1996, *Concurrent Engineering Fundamentals: Volume I*, Prentice Hall.

Reidsema, C. and Szczerbicki, E., 1997, Multi-Agent Systems for Concurrent Engineering. *Systems Analysis Modelling Simulation*, Vol. 28, pp. 257-279.

Rezayat, M., 2000a, The Enterprise-Web Portal for Life-Cycle Support. *Computer-Aided Design*, Vol. 32, pp. 85-96.

Rezayat, M., 2000b, Knowledge-Based Product Development Using XML and KCs. *Computer-Aided Design*, Vol. 32, pp. 299-309.

Roller, D. and Eck, O., 1999, Knowledge-Based Techniques for Product Databases. *International Journal of Vehicle Design*, Vol. 21, No. 2/3, pp. 243-265.

Roy, U. and Kodkani, S. S., 2000, Collaborative Product Conceptualization Tool Using Web Technology. *Computers in Industry*, Vol. 42, No. 2, pp. 195-209.

Roy, U., Bharadwaj, B., Kodkani, S. S., and Cargian, M., 1997, Product Development in a Collaborative Design Environment. *Concurrent Engineering Research and Applications*, Vol. 5, No. 4, pp. 347-365.

Seilonen, I., 1995, Data Modeling Issues in Product Management. *VTT Symposium 160: Product Models in Design and Production Planning*, (ed.), H. Johinen, Technical Research Centre of Finland, pp. 83-104.

Shah, J. J., 1989, Feature Transformations Between Application Specific Feature Spaces. *Computer-Aided Engineering Journal*, Vol. 5, No. 6, pp. 247-255.

Shah, J. J. and Mantyla, M., 1995, *Parametric and Feature-Based CAD/CAM*, John Wiley & Sons.

Shah, J. J. and Rogers, M. T., 1988, Functional Requirements and Conceptual Design of the Feature-Based Modeling System. *Computer-Aided Engineering Journal*, Vol. 5, No. 1, pp. 9-15.

Shen, W. and Barthes, J. P., 1995, DIDE: A Multi-Agent Environment for Engineering Design. *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, pp. 344-351.

Shen, W. and Norrie, D. H., 1999, Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems: An International Journal*, Vol. 1, No. 2, pp. 129-156.

Shen, W., Norrie, D. H., and Barthes, J. P., 2000, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor & Francis, London, UK.

Shen, W., Norrie, D. H., and Kremer, R., 1999, Developing Intelligent Manufacturing Systems Using Collaborative Agents. *IMS 99*, Leuven, Belgium.

Shi, Y. H., Eberthart, R. C., and Chen, Y. B., 1997, Design of Evolutionary Fuzzy Expert System. *Proceedings of 1997 Artificial Neural Networks in Engineering Conference*, St. Louis.

Shi, Y. H. and Eberthart, R. C., 1998, Parameter Selection in Particle Swarm Optimization. *The 7th Annual Conference on Evolutionary Programming*, San Diego.

Singh, N., 1995, *Systems Approach to Computer-Integrated Design and Manufacturing*, John Wiley & Sons.

Sriram, D. and Logcher, R., 1993, The MIT DICE Project. *IEEE Computer*, Vol. 26, No. 1, pp. 64-65.

Stonebraker, M., 1992, The Integration of Rule Systems and Database Systems. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 5, pp. 415-423.

Suh, N. P., 1990, *The Principle of Design*, Oxford University Press, New York.

Tan, S. T., Yuen, M. M. F., and Hui, K. C., 1987, Modeling Solids with Sweep Primitives. *Computers in Mechanical Engineering (CIME) Magazine*, September, pp. 60 -73.

Tso, S. K., Lau, H. C. W., Ho, J. K. L., and Zhang, W. J., 1999, A Framework for Developing Agent-Based Collaborative Service-Support System in a Manufacturing Information Network. *Engineering Application of Artificial Intelligence*, Vol. 12, pp. 43-57.

Ulieru, M., Norrie, D. H., Kremer, R., and Shen, W., 2000, A Multi-Resolution Collaborative Architecture for Web-Centric Global Manufacturing. *Information Science* (an Elsevier Journal) – Special Issue on Computational Intelligence for Manufacturing Applications.

Vickers, D. L. and Swanson, K. A., 1988, A Form Feature-Centered Architecture for Product Definition Exchange. *AUTOFACT '88 Conference Proceedings*, pp. (2-25) - (2-37).

Vliet, J. W., Luttervelt, C. A., and Kals, H. J. J., 1999, State-of-the-Art Report on Design for Manufacturing. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Waldron, M. B., Brown, D., and Yoshikawa, H. (eds.), 1992, *Intelligent Computer Aided Design*, North-Holland, Amsterdam.

Wu, J., 1999, *Distributed System Design*, CRC Press.

Xue, D., 1997, A Multilevel Optimization Approach Considering Product Realization Process Alternatives and Parameters for Improving Manufacturability. *Journal of Manufacturing Systems*, Vol. 16, No. 5, pp. 337-351.

Xue, D. and Dong, Z., 1993, Feature Modeling Incorporating Tolerance and Production Process for Concurrent Design. *Concurrent Engineering: Research and Applications*, Vol. 1, pp. 107-116.

Xue, D. and Dong, Z., 1994, Developing a Quantitative Intelligent System for Implementing Concurrent Engineering Design. *Journal of Intelligent Manufacturing*, Vol. 5, pp. 251-267.

Xue, D. and Dong, Z., 1997, Coding and Clustering of Design and Manufacturing Features for Concurrent Design. *Computers in Industry*, Vol. 34, pp. 139-153.

Xue, D., Rousseau, J. H., and Dong, Z., 1996, Joint Optimization of Performance and Costs in Integrated Concurrent Design: Tolerance Synthesis Part. *Engineering Design and Automation*, Vol. 2, No. 1, pp. 73-89.

Xue, D., Takeda, H., Kiriyama, T., Tomiyama, T., and Yoshikawa, H., 1992, An Intelligent Integrated Interactive CAD – A Preliminary Report. *Intelligent Computer Aided Design*, (eds.), M. B. Waldron, D. Brown, and H. Yoshikawa, North-Holland, Amsterdam, pp. 163-192.

Xue, D., Yadav, D., and Norrie, D. H., 1999, Knowledge Base and Database Representation for Intelligent Concurrent Design. *Computer-Aided Design*, Vol. 31, pp. 131-145.

Yadav, S., 1999, *Development of a Feature-Based Intelligent Design System*, A Master's Thesis, Department of Mechanical and Manufacturing Engineering, The University of Calgary.

Yoshikawa, H., 1988, Intelligent CAD. *Organization of Engineering Knowledge for Product Modeling in Computer Integrated Manufacturing*, (eds.), T. Sata, Elsevier, pp. 1-14.

Zhang, Y., Zhang, C, and Wang, H. P., 2000, Internet Based STEP Data Exchange Framework for Virtual Enterprises. *Computers in Industry*, Vol. 41, No. 1, pp. 51-63.