Investigation for a feature-oriented product design

Tang, Shaohui

2007

Tang, S. H. (2007). Investigation for a feature-oriented product design. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/36034>

<https://doi.org/10.32657/10356/36034>

a529491 177
3X
D/Blue

The Investigation For A Feature-oriented Product Database

Tang Shaohui

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2007

ABSTRACT

This research work is devoted to realizing a feature-oriented database system in order to support concurrent and collaborative engineering. The emphasis is on feature modeling in database so that information integration and sharing can be achieved. In the current globalized market with a virtual engineering environment, it is common for engineering tasks to be carried out by a group of engineers who are distributed in terms of both time and space. Furthermore, different engineering partners may use different application systems. A literature survey carried out in this thesis reveals that although many researchers have done much work to reduce the application gaps among different processes, in-depth research is still required to model, analyze and implement a system that can manage the information throughout different stages of product life cycles. In this research project, an original four-layer information model is proposed on the basis of an extended-STEP (standard for the exchange of product model data) framework. To realize this, a generic feature representation schema is developed. It is expected that the proposed information model can be the repository of the entire product model (EPM) that covers the information of a product's entire life cycle. The central piece of this infrastructure is the unified conceptual schema for the feature-oriented database. Sub-models, corresponding to external schema of different applications can be generated to provide specific views of the EPM. Mapping mechanisms between EXPRESS-defined information model and database schemas are also investigated. Based on these mechanisms a feature-oriented database is designed. To manage product information, the mechanisms for feature model evaluation and validation are investigated. A prototyped feature-oriented database system is established with the aforementioned approaches. By integrating this prototyped database with web technology, the information integration in a global collaborative environment can be achieved.

ACKNOWLEDGEMENTS

I am here to express my sincere appreciations to those in guiding and supporting my accomplishment of the work.

First and foremost, I would like to express my deep gratitude to my supervisor, A/P Ma Yongsheng. He has given me excellent supervision, invaluable advice and continuous attention throughout my research. Under his edifying and enlightening guidance, I have saved a lot effort from straying around and focused straight on the research issues.

I am in debt to NTU for awarding me with a research scholarship.

I appreciate all the help, encouragement, advice and cheers from my friends in my life, especially from Mr. Chen Gang for his continuous help. Besides, I would like to thank all the technical staff of Design Research Center and CAD/CAM Lab for their support to my project.

Last but not least, I wish to thank my family for their understanding, patience, encouragement, and continuous moral support.

Thank you all.

LIST OF PUBLICATIONS

Tang, S. H., Ma, Y. S. and Chen, G., "A feature-oriented database framework for web-based CAX applications", *Computer-Aided Design & Applications*, Vol. 1, pp. 117-125, 2004.

Tang, S. H., Ma, Y. S. and Chen, G., "A Web-based Collaborative Feature Modeling System Framework", *Proceedings of the 34th International MATADOR conference*, pp. 31-36, 2004.

Tang, S. H., Ma, Y. S. and Chen, G., "Integration of a Solid Modeler with a Feature-oriented Database", presented on CAD'05, conference, Bangkok, Thailand, June 20-24, 2005.

Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Unified feature modeling scheme for the integration of CAD and CAX", *Computer-Aided Design & Applications*, Vol. 1, pp 595-602, 2004.

Ma, Y. S., Britton, G. A., Tor, S. B., Jin, L. Y., Chen, G. and Tang, S. H., "Design of an feature-object-based mechanical assembly library", *Computer-Aided Design & Applications*, Vol. 1, pp 379-403, 2004.

Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Unified feature based integration of design and process planning", *Proceedings of the 34th International MATADOR conference*, pp. 63-68, 2004.

Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Knowledge-Based Reasoning in a Unified Feature Modeling Scheme", *Computer-Aided Design & Applications*, Vol. 2 pp. 173-182, 2005.

Chen, G., Ma, Y. S., Ming, X. G., Thimm, G., Lee, S. S. G., Khoo, L. P., Tang, S. H., and Lu, W. F., "A unified feature modeling scheme for multi-applications in PLM", *proceedings of The 12th ISPE International Conference on Concurrent Engineering (CE2005): Research and Applications -Next Generation Concurrent Engineering: Smart and Concurrent Integration of Product Data, Services, and Control Strategies*, M. Sobolewski and P. Ghodous (Eds.), ISPE, Dallas, pp. 343-348, 2005.

TABLE OF CONTENTS

ABSTRACTI

ACKNOWLEDGEMENTSII

LIST OF PUBLICATIONSIII

TABLE OF CONTENTSIV

LIST OF FIGURESX

LIST OF TABLESXIV

LIST OF ABBREVIATIONSXV

CHAPTER 1 INTRODUCTION1

1.1 Problem Statements..... 1

1.2 Objectives 4

1.3 Research Approach 4

1.4 Scope 5

1.5 Original Contributions of This Research 8

1.6 Organization of the Thesis 8

CHAPTER 2 LITERATURE REVIEW 10

2.1 Product Design and Development 10

2.1.1 Overview of Product design and Development 10

2.1.2 Concurrent and Collaborative Engineering 10

2.2 Product Data Exchange and Some Industrial Standards12

2.3.1 IGES 12

2.2.2 VDAFS and SET 13

2.2.3 STEP	13
2.2.3.1 Description Method	14
2.2.3.2 STEP Information Models	15
2.2.3.3 Implementation Methods	16
2.2.3.4 Conformance Testing	17
2.3 Information Exchange (Sharing)	17
2.3.1 Information Integration Trend	17
2.3.1.1 Objectives of Information Sharing	18
2.3.1.2 Requirements of Information Sharing	18
2.3.2 Data Exchange among Applications	19
2.3.3 Research Effort for Multi-application Information Sharing	20
2.3.3.1 Neutral File – Level 1	21
2.3.3.2 Standard Data Access Interface (SDAI) – Level 2	21
2.3.3.3 Database – Level 3	23
2.3.3.4 Knowledge Base – Level 4	28
2.3.4 Commercial Effort	28
2.3.4.1 CAD Web Portals	28
2.3.4.2 Groupware Tools	29
2.3.5 Integration Platform	30
2.3.5.1 Computer Network-Based Integration	30
2.3.5.2 CORBA	32
2.3.6 Database Technology	33
2.3.6.1 DataBase Management System (DBMS)	33
2.3.6.2 Types of Database	33
2.3.6.3 Architecture of DBMS	36

2.3.6.4 Database Schema 37

2.4 Feature Technology 38

2.4.1 Feature Definition 38

2.4.2 Feature Information Sharing40

2.4.3 Constraint Solving 42

2.4.4.1 Constraint Classification 42

2.4.4.2 Constraint Solving 42

2.4.4 Geometrical Modeling Kernel 44

2.4.4.1 Overview 44

2.4.4.2 ACIS 3D Geometrical Modeler (Spatial/Dassault systems) 45

2.5 Technological gaps 46

2.6 Summary 49

CHAPTER 3 INFORMATION INTEGRATION INFRASTRUCTURE AND
GENERIC FEATURE MODEL 50

3.1 Introduction 50

3.2 Four-layer Information Integration Infrastructure50

3.3 Generic Feature Model 54

3.3.1 Feature Shape Representation 55

3.3.2 Validity Condition (Constraint) Definition 58

3.3.3 Other Generic Feature Properties 62

3.3.4 Member Functions 64

3.4 Application-specific Feature Model 66

3.4.1 Design Feature Representation66

3.4.1.1 Design Feature Representation Schema 66

3.4.1.2 An Example Design Feature Definition - Slot 68

3.4.2 Manufacturing Feature Representation..... 74

 3.4.2.1 Manufacturing Feature Representation Schema 74

 3.4.2.2 An Example Machining Feature Definition – Slot 78

3.5 Operation for Multi-application Interoperability 81

 3.5.1 Multi-application Interoperability 81

 3.5.2 Operation 83

3.6 Geometrical Representation Schema — Cellular Model 85

 3.6.1 *Cellular Model* 85

 3.6.2 Boolean Operation 86

3.7 Summary 87

CHAPTER 4 DESIGN OF FEATURE-ORIENTED DATABASE 88

4.1 Introduction 88

4.2 Mapping Mechanisms 88

 4.2.1 Mapping to Database Schemas 89

 4.2.2 Mapping to Workform Format 91

 4.2.2.1 Geometry mapping 92

 4.2.2.2 Generic feature definition under ACIS framework 92

4.3 Schema Definition for Feature-oriented Database 95

 4.3.1 Geometrical Representation in Database 96

 4.3.2 Generic Feature Representation in Database 99

 4.3.3 Product and Part representation in database101

4.4 A Simple Case Study102

4.5 Summary107

CHAPTER 5 FEATURE MODEL EVALUATION AND VALIDATION .108

5.1 Introduction108

5.2 Feature Model Evaluation108

 5.2.1 Overview108

 5.2.2 Dynamic Maintenance of Feature Precedence Relations.....111

 5.2.3 History-independent Feature Model Re-evaluation118

 5.2.3.1 Adding a New Feature Instance 118

 5.2.3.2 Deleting a Feature Instance 121

 5.2.3.3 Modifying a Feature Instance 122

 5.2.3.4 B-rep Evaluation123

 5.2.3.5 A Case Study 124

5.3 Maintenance of Feature Validity 131

 5.3.1 Validity of Feature Geometry 131

 5.3.2 Constraint Solving 132

 5.3.2.1 Local Constraint Solving 132

 5.3.2.2 Multi-view Constraint Propagation 137

 5.3.2.3 Constraint Conflict management 141

5.4 Persistent Naming141

5.5 Summary142

CHAPTER 6 A FEATURE-ORIENTED DATABASE FOR A MULTI-VIEW
AND WEB-BASED MODELING SYSTEM 144

6.1 Introduction 144

6.2 System Design 144

 6.2.1 System Requirements 144

 6.2.2 System Framework 145

 6.2.2.1 Web Server 146

 6.2.2.2 Application Object Server 146

<i>Ph. D Thesis</i>	<i>Table of Contents</i>
6.2.2.3 Feature Object Server	147
6.2.2.4 Database Server	148
6.2.2.5 Geometrical Modeler	148
6.2.3 Information Flow	150
6.3 Prototype the Proposed System	150
6.3.1 The Case	150
6.3.2 Feature-oriented Database	151
6.3.3 Database Management	154
6.3.3.1 Save Algorithm	156
6.3.3.2 Restore Algorithm	157
6.3.4 Data Concurrency and Security Control	159
6.3.4.1 Concurrency Control	159
6.3.4.2 Security Control	161
6.3.5 Web-based Multi-client Collaboration	164
6.3.6 Web-based Access	168
6.4 Summary	171
CHAPTER 7 CONCLUSIONS AND FUTURE WORK	173
7.1 Conclusions	173
7.2 Contributions	174
7.3 Limitations	177
7.4 Future work	178
REFERENCES	180

LIST OF FIGURES

Figure 1.1 Organization of the thesis 9

Figure 2.1 Figure 2.1 STEP structure 14

Figure 2.2 The SDAI data architecture17

Figure 2.3 Proprietary direct translators 19

Figure 2.4 Neutral intermediate format 20

Figure 2.5 Levels of multi-application information sharing 20

Figure 2.6 Basic architecture of CORBA 32

Figure 2.7 Database architecture (two-tier) 36

Figure 2.8 Database Architecture (Three-tier)37

Figure 2.9 Three-level schema DBMS architecture 38

Figure 3.1 Four-layer information infrastructure51

Figure 3.2 Partial schema-level EPM52

Figure 3.3 Figure 3.3 Unified feature model..... 53

Figure 3.4 Symbol convention of EXPRESS-G 54

Figure 3.5 Generic feature representation schema 55

Figure 3.6 Block feature 56

Figure 3.7 Other primitive features with parameters 56

Figure 3.8 Constraint representation schema 59

Figure 3.9 An example for semantic constraints 59

Figure 3.10 Constraint_strength for constraint solving 61

Figure 3.11 Example part 63

Figure 3.12 Constraint graph and Feature dependency graph of example part 63

Figure 3.13 Design feature representation schema	67
Figure 3.14 Slot feature definition in EXPRESS-G	68
Figure 3.15 Path definition in EXPRESS-G	69
Figure 3.16 Slot end type with parameters	69
Figure 3.17 Open_profile definition in EXPRESS-G	70
Figure 3.18 Square_U profile with parameters	70
Figure 3.19 An open ended 'slot' design feature with the square U profile	71
Figure 3.20 A design feature (slot) instance	73
Figure 3.21 Manufacturing feature representation schema	74
Figure 3.22 An Example of compound_feature	75
Figure 3.23 Multi_axis_feature representation	76
Figure 3.24 Machining_operation schema	77
Figure 3.25 Accessing and removal volumes of machining features	78
Figure 3.26 Machining feature (slot) definition in EXPRESS-G	80
Figure 3.27 The instance of a machining slot feature	81
Figure 3.28 Operation representation schema	84
Figure 3.29 Partial geometry schema with cellular topology	85
Figure 4.1 Information mapping structure	88
Figure 4.2 Feature derivation structure in EXPRESS-G	93
Figure 4.3 Generic feature definition with ACIS entities	95
Figure 4.4 Partial database schemas for geometrical representation	96
Figure 4.5 Linked list for ordered list	98
Figure 4.6 Generic schema for non-ordered list	99

Figure 4.7 Generic feature representation in database	100
Figure 4.8 Product representation in database	101
Figure 4.9 A case part	102
Figure 4.10 Cellular model of the example part	104
Figure 4.11 Final part geometry	104
Figure 4.12 Database representation of the case part	105
Figure 5.1 Example part	109
Figure 5.2 The cellular model	110
Figure 5.3 The B-rep model resulted from the history-based evaluation	110
Figure 5.4 The expected B-rep model	111
Figure 5.5 Feature precedent relation determination	112
Figure 5.6 Multiple depended features	114
Figure 5.7 The feature dependency graph	114
Figure 5.8 Two independent features with no overlapping	115
Figure 5.9 Two independent overlapping features with the same nature	116
Figure 5.10 Algorithm for precedence order generation	117
Figure 5.11 Creation of slot feature on the base block	120
Figure 5.12 Feature deletion	122
Figure 5.13 Boundary detection	124
Figure 5.14 Result of B-rep evaluation	124
Figure 5.15 A case part	125
Figure 5.16 Dependency graph of the initial case part	127
Figure 5.17 After adding a boss for case I	128
Figure 5.18 Dependency graph for case I	129
Figure 5.19 After adding a boss for case II	129

Figure 5.20 Feature dependency graph for case II	130
Figure 5.21 After removing the boss feature	130
Figure 5.22 Feature dependency graph after removing boss feature	131
Figure 5.23 Maintenance of feature validity	132
Figure 5.24 Constraint solving procedure	133
Figure 5.25 Constraint graph of the example part	135
Figure 5.26 Updated constraint graph for the example part	136
Figure 5.27 Different views linked by inter-application constraints	140
Figure 6.1 Overall system architecture	147
Figure 6.2 Integration of modeler with the proposed system	149
Figure 6.3 Assembly model for the vise product	151
Figure 6.4 The database tables created in the prototype	152
Figure 6.5 Partial geometrical data stored in database	153
Figure 6.6 The feature list of fix_jaw part	153
Figure 6.7 Constraint graph of fix_jaw in design view	154
Figure 6.8 Save() algorithm	156
Figure 6.9 Restore() algorithm.....	158
Figure 6.10 Registration of a collaboration session with the session manager	164
Figure 6.11 Design features of fix_jaw	165
Figure 6.12 Send feature operation from the edit view client	167
Figure 6.13 Receive feature operation and update the model in another design view.....	167
Figure 6.14 The manufacturing features generated	169
Figure 6.15 MDAI security management	170
Figure 6.16 Product information check-in through MDAI	170
Figure 6.17 Product information check-out through MDAI	171

LIST OF TABLES

Table 3.1 Other features with parameters 57

Table 4.1 Mapping of primitive data type 89

Table 4.2 Example of inheritance mapping 90

Table 4.3 Example of enumeration mapping90

Table 4.4 Example of selects mapping91

Table 4.5 Example of aggregate data type mapping91

Table 5.1 Features, constraints and parameters in the example part126

Table 5.2 Redefine two features128

Table 5.3 Inter-application constraint table138

LIST OF ABBREVIATIONS

AAG	attributed adjacency graph
ADT	abstract data type
AICs	application integrated constructs
AP	application protocol
API	application programming interface
ARM	application reference model
Brep	boundary representation
CE	concurrent engineering
CAD	computer-aided design
CAE	computer-aided engineering
CAM	computer-aided manufacturing
CAPP	computer-aided process planning
CAX	computer-aided x application
CIMS	computer integrated manufacturing system
CORBA	common object request broker architecture
CSG	constructive solid geometry
DBA	database administrator
DBMS	database management system
DDL	data definition language
DML	data manipulation language
DTD	document type declaration
DWG	drawing

ECO	engineering change order
EDM	engineering data management
ERP	enterprise resource planning
EPM	entire product model
ESPRIT	European strategic program for research and development in information technology
HTML	hypertext markup language
IDL	interface definition language
IGES	initial graphics exchange specification
IR	integrated resource
ISO	international standard organization
JDBC	java database connectivity
KQML	knowledge query and manipulation language
LAN	local area network
LDF	longest depended feature
LLP	longest length of path
MADE	manufacturing and automation design engineering
MDAI	multiview data access interface
MRP	manufacturing resource planning
OCCI	ORACLE C++ call interface
ODBC	open database connectivity
OID	object identifier
OODBMS	object-oriented database management system
ORDBMS	object-relational database management system
PDES	product data exchange standard

PDM	product data management
PL/SQL	procedure language/structured query language
QDM	quiet data management
RDBMS	relational database management system
SDAI	standard data access interface
SQL	structured query language
STEP	standard for the exchange of product model data
UI	user interface
VRML	virtual reality markup language
WAN	wide area network
WWW	world wide web
XML	extensible markup language

CHAPTER 1 INTRODUCTION

1.1 Problem Statements

Today's market is under rapid and continuous change. With the globalization of multinational corporations, the competition in the global market has been greatly intensified. This pressure enforces companies to develop product with the highest quality, at the lowest price, and more importantly, in the shortest time-to-market [Clark 1987, Dumaine 1989, Gonzalez 2002 and Fynes 2005]. More importantly, because shortening time-to-market will extend the effective product selling period, increase market share and achieve price priority [Stalk 1988], therefore, it is the critical objective for many companies [Terwiesch 1999, Mahmoud-Jouini 2004 and Nwagboso 2004].

In order to speed up product development processes, the concepts of concurrent engineering and collaborative engineering occurred. In a concurrent and collaborative engineering environment, many engineers with diverse skills, expertise, temperaments and personalities are involved and are responsible for different tasks. Vast amount of knowledge and information involved in product development is far more than any individual can manage. It can be appreciated that, instead of managing the information via each application system in separated data formats, a database management system (DBMS) can be effectively used to manage all the product information concurrently in a consistent manner in order to eliminate duplicated data. For a well-defined database, the methods for reading, writing, updating and deleting operations are generic. A DBMS can also provide shared user-accesses as well as the mechanisms to ensure the security and integrity of the stored data.

Currently, many computer-aided software tools have been incorporated into product development processes, which include Computer Aided Design (CAD), Computer Aided Process Planning (CAPP), Computer Aided Engineering (CAE) and Computer Aided

Manufacturing (CAM) tools. Traditionally, such software tools are regarded as individual applications. This means the data generated and used by one application system can not be used directly by another one because they use their native data formats. Two strategies for data exchange and sharing were proposed: developing proprietary direct translators, or making use of a public domain neutral intermediate (database) format [Owen1993]. With the first strategy, a translator is needed between every two systems. Although a direct translator has the advantages of being fast and accurate, it is only suitable for the virtual enterprise that contains few participating systems. For a number of collaborative partners with different systems, it will be prohibitive to develop the data translators between each pair of them. Comparing with the first strategy, the second strategy provides a neutral data format which allows the information sharing via a data access interface, and is more practical for implementation in a collaborative engineering environment.

The most important issue for information sharing is the consistent product model. The existing standards for data exchange include IGES, VDAFS, SET, STEP and others. STEP is the most popular one. It defines the information model in EXPRESS language, which is human readable and computer recognized. STEP provides a series of application protocols (APs) for different specific application areas, covering the information model throughout a product's entire life cycle. In fact, STEP has been increasingly accepted by both CAD vendors and users. However, simply adopting STEP is not appropriate because there are still a lot of technical issues to be resolved, such as those related to feature semantics and constraints.

As discussed in the next chapter, the literature review, it appears that the following issues still exist so far:

- *Duplicated data and conflicts*

Product and process information is often stored in a file format, which means duplicated data and potential conflicts. In addition, files are not flexible enough to support the multi-view functions required by different applications. Furthermore, efficiently and effectively indexing distributed files and extracting useful information from files are difficult. For example, multiple end-users or applications can neither cross-refer the data created; nor synchronize definitions and modifications easily.

- *Information loss*

As to product information, although many existing systems claim supporting CAD-neutral, STEP-based data exchange, they lack feature level interoperability. Useful information regarding to features and other semantic properties, is often stripped off during data export and import processes. Therefore complete information sharing has not been achieved.

- *The absence of a generic feature definition*

Most of the database designs lack of a higher-level generic feature definition, which can be possibly used as a common template for database schema extension in the future; therefore, these proposed databases suffer from the problem of scalability.

- *No explicit schema and efficient mechanisms for engineering databases*

Although there are a few research works covering the construction of engineering databases to support collaborative product development, however, most of them focus on high-level structural explanation of product model [Domazet 2000, da Cunha 2002, Wang 2002, 2003 and Lee 2005] without offering explicit database schema definitions. In addition, most of them did not provide efficient and effective mechanisms to support product information management in a consistent and concurrent manner. Last but not least,

without high-level information integration infrastructure, their works have not solved the problem of engineering collaboration.

1.2 Objectives

This research work aims at establishing the effective mechanisms for designing and implementing a feature-oriented database to support collaborative product development. The detailed objectives are as follows:

Objective 1: *To investigate and design an information integration infrastructure that can support the whole product life cycle with different applications;*

Objective 2: *To investigate and define a neutral intermediate data format (including feature model and geometric model) for information sharing;*

Objective 3: *To propose and prototype a database that provides the repository for information management and sharing; and*

Objective 4: *To develop a prototype system so as to test the functionality of the proposed database design.*

1.3 Research Approach

The research approach used in this work can be justified as associative and unified feature-based database technology towards information integration for collaborative product development. The characteristics are discussed in more details as follows:

- **Associative and unified features**

As to be reviewed in chapter 2, most of the current software tools for product development are already feature-based but limited to individual applications. In this work, the proposed information integration infrastructure supports multi-stage applications throughout product lifecycles, because a unified feature modeling scheme is adopted. This infrastructure covers not only a generic feature representation schema, but also the associative feature constraint management and the implementation methods for the product information repository. Therefore, high-level feature information integration is supported.

- **Feature-oriented database technology**

It has been recognized that database approach is more efficient than file-based approach [Mittra 1991, Fortier 1997 and Ramakrishnan 2000]. Feature-oriented database is generic for information adding, deleting, updating and querying. Furthermore, database is expected to reduce duplicated data and potential data inconsistency. Ideally, in a feature-oriented database, product and process information across different associated database tables, which may include geometrical entities, features, security data and others, is globally managed and shared for multi-users via the network. Based on this feature-oriented database technology, knowledge workers or application software developers can consistently reorganize the product model for different application views; at the same time the product model repository can be constructed with levels of granularity. Such a repository system is expected to enhance the support for concurrent and collaborative engineering.

1.4 Scope

This work is limited to the mechanical product design phase, which covers product design and manufacturing analysis. The generic feature model proposed here could be

expanded in order to apply for other CAx applications or applications in other engineering domains such as civil engineering or electrical and electronic product design. The scope of the research includes:

- *The design of an information integration infrastructure.*

The proposed information integration infrastructure is intended to provide high-level guidelines for information integration for collaborative product development. The rest of this research work is conducted under this infrastructure.

- *The definition of a generic feature representation schema on the basis of STEP.*

The generic feature model covers feature shape representation schema and constraint representation schema. In the feature definition, all different kinds of constraints are regarded as associative attributes attached to geometrical entities and features. In this work, the information model is described using EXPRESS language. All the symbols follow the convention of EXPRESS language definitions. Note that the classification of application features (design and manufacturing features) does not accommodate all types of features because the feature types are not limited. But due to the extensibility of generic feature model, all types of features can be defined.

- *The methods for feature model evaluation and validation based on feature semantics.*

The proposed approach is to dynamically maintain the feature dependency graph (constraint graph) and properly propagating feature labels. It aims at keeping feature data consistency and association, and solving semantic problems suffered by most current CAD systems. In this work, we adopt some existing techniques for constraint solving and persistent naming. Some outstanding problems in constraint solving and persistent naming are not focused.

- *The investigation on the mapping mechanisms from EXPRESS-defined information model to the workform format.*

In order to support feature-oriented database, a geometrical modeling kernel is incorporated to provide low-level geometrical modeling service such that high-level feature modeling can be realized. On the basis of newly developed mapping mechanisms, database runtime model can be supported and feature-oriented database can be achieved. As ACIS is adopted as the geometrical modeling kernel for this work, workform format mentioned hereafter is specific to ACIS data structure.

- *The design and implementation methods for a feature-oriented database to support collaborative product development.*

Feature-oriented database can provide physical repository for the management of product and process information. Therefore, this database covers not only geometry data, but also feature information, such that high-level feature-based information integration and sharing can be achieved. In this work, only design and manufacturing features are considered. The cellular model is adopted as the geometrical representation schema as described in ACIS [ACIS 2001]. In addition, ORACLE, an object-relational database is used in the work. Theoretically, the geometric representation schema generated in this work can be equally implemented with other major commercial databases such as DB2 and Informix.

- *The prototype development for demonstrating a database-driven, network-based feature modeling system on the basis of the proposed methodology.*

In order to illustrate the whole work done in this research, a feature-oriented database is designed and implemented. Design application and manufacturing application are used to test the functionality of database design.

1.5 Original Contributions of This Research

Compared with the reported works in the literature as to be reviewed in Chapter 2, this research is towards to solve the interoperability of CAD systems at feature level. This approach is novel because traditionally, interoperability is difficult to be realized and all research and development effort is carried out largely in a file-based approach and representing only geometry. The original contributions of this research are as follows:

- A common generic feature definition supporting different computer aided applications is developed. It encapsulates associated geometric and non-geometric entities and is flexible and scalable such that product information is represented in object-oriented feature entities. Such feature objects support polymorphism and make the information sharing possible with finer level of granularity in comparison with the traditional file-based approach. On top of this generic feature definition, a new concept – “feature operation” is proposed and implemented to achieve efficient communication between multiple CAD systems without heavy data load.
- Feature level information is represented and manipulated in the database. The database is designed as feature-oriented by using the proposed mapping mechanisms from the neutral format product model. Therefore, feature level information as well as the associations among them can be persistently kept and managed.
- The database design is natural for collaborative engineering compared with traditional file-based information sharing. A database is generic for reading, writing, updating and deleting operations across the network and therefore is suitable for multi-user and multi-application collaboration as well as information integration.
- The prototype system proves that a geometric modeler can be integrated with the feature-oriented database to provide lower-level geometrical modeling services, on the basis of which higher-level feature modeling, evaluation and functional services can be realized.

1.6 Organization of the Thesis

The thesis is organized into seven chapters. This chapter is the project introduction, covering the technological background and the trend, the research approach, objectives, significances and the scope. Chapter 2 provides a systematic review of the work and techniques relevant to the proposed study. Chapter 3 proposes a design of four-layer

information integration infrastructure within which, the unified feature model and generic feature representation schema are presented. Chapter 4 investigates the mapping mechanisms from EXPRESS-defined information model to the ACIS workform format, and further to the designed database schemas. The generic feature representation is mapped to the workform format. Then, the design for a feature-oriented database is presented. Chapter 5 describes a history-independent approach for feature model evaluation to keep feature consistency in the database. This is achieved by dynamically maintaining the feature dependency graph and propagating feature labels during modeling operations. In addition, constraint solving mechanisms are illustrated. Chapter 6 proposes a web-based feature-oriented database system and implemented a prototype system to support such a typical working scenario for collaborative product development. Chapter 7 summarizes the research and draws the conclusions. The main contributions achieved in this work are listed. In addition, some suggestions for future work are described. The organization of the thesis is shown in Figure 1.1.

Figure 1.1 Organization of the thesis

CHAPTER 2 LITERATURE REVIEW

This chapter starts with reviewing the basic technologies applied in product design and development including concurrent and collaborative engineering. Next, the issue of interoperability and the existing standards for product data exchange are discussed. After that, the strategies for information sharing among CAX applications are screened, where database and web technologies are also examined. Then followed is the review on the current state of art of the feature technology because the proposed information repository is feature-oriented. Finally, the technological gaps are identified; the required research areas are highlighted.

2.1 Product Design and Development

2.1.1 Overview of Product Design and Development

Product development determines the long-term viability of companies and even the related economies [Ulrich 1995 and Smith 1999]. Generally, it consists of several phases ranging from the analysis of customer requirements to the adoption of the product by manufacturing, namely, conceptual design, technical design, detail design, and manufacturing process design [Dixon 1990 and Mallick 1992].

During the past decades computer-based tools have been introduced in industry to perform technical tasks such as drafting, design, process planning, data acquisition, process control, and quality assurance. However, all these solutions are, in most cases, developed independently; and each of them focuses only on one stage of the product life cycle.

2.1.2 Concurrent and Collaborative Engineering

Traditionally, product development processes are organized in a “serial” approach, i.e. starting with product design; CAE analysis, process planning and manufacturing were

executed one by one until the final product entered the market. Correspondingly, the information flow based on this serial layout is often adopted and referred to as “throwing it over the wall” strategy as each group of specialists does their work and then passes it on to the next group. This approach has many shortcomings, such as higher costs and longer development cycles due to unavoidable iterative backtrackings and modifications [Chang 1998].

In 1980s, the intensive competition in global market requires companies to develop product with the highest quality, at the lowest price and most importantly, the shortest time to market; then Concurrent Engineering (CE), was introduced. Soon, CE was recognized as the systematic approach to achieve the integrated, concurrent design of products and their related processes, including manufacturing and support [Carter 1992, Prasad 1996]. This approach requires the developers to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule, user requirements [Pham 1998], and even those suppliers from the outset of the product development cycle. The objective of CE is to produce a model of robust design that is tolerant of manufacturing and use variation, at the lower cost than sequential approach and offer the potential for faster development of higher quality, more producible product through a better integration of activities and processes. By paying attention to all aspects of the design at each phase, errors are detected prior to being implemented in the production. The integrated design process includes a strong information sharing management, iterative processes of redesigns and modifications, knowledge-based trade-off analyses for design optimization, and the documentation of all parts of the design.

More recently, collaborative product development becomes the major trend because of the globalization of economies. It is the application of team-collaboration practice to an organization’s product development endeavors. It is built on the nature of cross-functional

product development teams introduced in the realm of concurrent engineering [Mills 1998], with the concept of highly effective and well-supported team collaboration. As such, collaborative engineering is an evolution of the principles and practices of concurrent engineering and therefore is regarded as the logical extension to concurrent engineering [Raper 2000].

Although concurrent and collaborative engineering can offer substantial benefits, which include the reductions of cycle time and cost with improved quality, and hence, many companies are beginning to take a strong interest in the collaborative approach, it is not yet clear how it can best be implemented. More specifically, the information modeling domain of concurrent and collaborative engineering encompasses a wide variety of aspects, from user interfaces to computer network management, and from information model to data storage.

2.2 Product Data Exchange and Some Industrial Standards

In order to enable interoperability among different applications using neutral intermediate format, a commonly acceptable, comprehensive and well-defined information model is crucial. A lot of efforts have been made to model product entire life cycle for the implementation of CE. One of the major efforts so far is to overcome the interoperability difficulty via data exchange. In this section, the existing standards are reviewed. The existing standards for data exchange (information sharing) include IGES, VDAFS, SET, and STEP.

2.2.1 IGES

IGES (Initial Graphics Exchange Specification) was the first specification for CAD data exchange [Smith 1988, Everitt 1988, and Kent 1990]. Although it has been widely

supported by most of the current commercial CAD packages, it was not designed for feature level information sharing.

2.2.2 VDAFS and SET

Several data exchange specifications have been worked out to overcome the deficiencies in IGES and to solve dedicated problems for specific industrial sectors. The two most well known standards are the German VDAFS and the French SET.

VDAFS was developed primarily for German car manufacturers to transfer free-form surfaces among different CAD/CAM systems. It was employed by and logged as a national standard in Germany (DIN 66301). Compared with IGES, VDAFS uses fewer types of entities which result in less errors. On the other hand, when converting a file from a native CAD format into VDAFS, a lot of information such as layers dimensions, 2D geometry and text are lost.

SET became an official French standard, Afnor Z68-300, SET, in 1985. It has been mainly employed by the European Aerospace industry. It is similar to IGES except that it features more general data structures than IGES in a more compressed file structure [Burton 1998].

2.2.3 STEP

International Standard Organization (ISO) has been working on its STEP (standard for the exchange of product model data) effort since 1984, and later, it was published as ISO 10303. By 1994, STEP has contained information, from design to analysis, manufacture, quality control testing, inspection, and product support functions [ISO1994a].

Basically, STEP covers geometry, topology, tolerances, relationships, attributes, assemblies, configuration, etc. STEP has been divided into multiple parts to achieve this ambitious goal. Figure 2.1 shows the structure of the STEP components. STEP contains

five main infrastructure parts which are description methods, integrated resource, implementation methods, application protocol and conformance testing.

Figure 2.1 STEP structure

STEP separates the techniques of representation of product information from the implementation methods. This design makes it possible that the STEP effort will not be out of date due to the fast development of computer technology. It specifies the implementation methods that support the exchange of product data defined in application protocols, as well as a formal data specification language EXPRESS, which is used to specify the representation of product information.

2.2.3.1 Description Method

EXPRESS is a kind of data specification language defined in part 11 of ISO 10303. It conforms to a formal syntax and can be processed by computer software [Owen 1993]. Therefore, it is human readable and computer recognized. EXPRESS is not a programming language like C, nor is it a functional interface description language like COBRA/IDL. It has a graphical form, EXPRESS-G, and an instance form, EXPRESS-I. EXPRESS has six declarative constructs, which are type declaration, entity declaration, schema, constant, algorithms and rule [ISO1994b]. EXPRESS provides a rich collection of data types and

inheritance organizations to describe product information. It uses supertype and subtype mechanism to enable the inheritance of data and constraints which has analogies in object-oriented programming and database management systems. Entities represent real-world objects and can be organized into complex inheritance graphs. There are simple data types such as real, integer, and string while other constructed data types are used to capture the meaning behind simple values, e.g. subtype enumeration and select are some of the varieties for aggregation. It also supports unions of different types. EXPRESS can describe complex functional and algorithmic constraints. There are ways to capture existence dependencies, keys, optional values, derived values, constraints on relationships, instances and the entire information base. The language contains a rich set of expressions, structured programming constructs, and a library of built in functions. For more details, please refer to [ISO1994b] and [Schenck 1994].

2.2.3.2 STEP Information Models

The most important issue in data exchange or sharing is the building of information models. STEP information models are described in EXPRESS. There are three classes, i.e. integrated resources (IRs), application protocols (APs) and application integrated constructs (AICs).

IRs are the kernel of STEP, these conceptual schemas are the resources on which all APs are built. There are two types of IRs: generic and application oriented. Generic integrated resources, designated as 40-series documents, describe very general characteristics of products across all industries. The application integrated resources, designated as 100-series documents, refine the integrated resources down to the needs of a particular industry. Some of these have reached international standard status while others are still in various stages of development.

Application Protocols are formal documents describing the information models for specific activities in the life cycle of a product on the basis of IRs. Each AP defines a set of activities, their information requirements within the scope, and a formal schema for these requirements that is tied to the integrated product model, which is shared among all APs. STEP leaves these APs open-ended for future expansion. APs are designated as the 200-series documents.

AICs are the sharing information model that can be used by different application protocols. The constructs are sets of refined definitions that must be used as a single unit, without any additional refinements. These AICs enable application-interpreted models to be built more quickly.

2.2.3.3 Implementation Methods

STEP defines implementation methods to enable data exchange. The first implementation method defined is a neutral file exchange using straightforward ASCII format for exchanging EXPRESS-defined data sets (Part 21 [ISO 1994c]).

Another implementation method defined in STEP is called the SDAI. The goal of this protocol is to reduce the cost of integrated product databases by making complex engineering applications portable across database implementations [Loffredo 1998]. The SDAI is described by several ISO standards documents. STEP Part 22 [ISO 1995a] contains a functional description of the SDAI operations. Figure 2.2 shows the SDAI data architecture and explains the relationship between application data, dictionary, session and population organizing data, the application program and the SDAI implementation. STEP also describes how to implement SDAI model with the C++ and C language environments in Parts 23 [ISO 1995b] and 24 [ISO 1995c] using an approach called language binding. Bindings for CORBA/IDL and Java are also under development.

Figure 2.2 SDAI data architecture [ISO 1995a]

2.2.3.4 Conformance Testing

Many standards such as IGES, SET, VDAFS suffer from the lack of conformance testing service when the standard was published. STEP provides a conformance testing framework and methodology, which are assigned as 30 series.

2.3 Information Exchange/Sharing

2.3.1 Information Integration Trend

Information system technology has long promised to integrate product design activities. It could seamlessly enable members of a team to exchange or share data and information across organizational and geographical boundaries. The reality, however, has been very different. Product data is often stored and duplicated in many different places. Commonly

the data files or databases are tightly bound to individual applications. Therefore, so far, there isn't a single information system which can manage all the information for collaborative product development.

2.3.1.1 Objectives of Information Sharing

Information sharing is the prerequisite for the implementation of concurrent and collaborative engineering. The need to share information between different engineering applications has long been recognized [Griffin 1995]. In a collaborative engineering environment, it is common for engineering tasks to be carried out by a group of engineers who may be distributed in terms of both time and space; furthermore, different engineering partners need to use different applications. This practice means a product model generated from an application system has to be shared directly by other ones. Currently, information sharing among CAx applications becomes the bottleneck for collaborative engineering.

2.3.1.2 Requirements of Information Sharing

The requirements for collaborative engineering are adapted from [Kang 1997] and extended as follows:

- Such a system should use international standard, since such standard allows participation by a wide variety of parties, including suppliers and customers.
- Translators should be used to communicate applications with a group of unified engineering databases.
- A DBMS should be used to manage the storage and retrieval of the large amount of information for a reasonable sized company. Such a DBMS can also readily allow for the integration with other functions within a company or across companies.
- Interactive 3-D graphical representations of parts and products should be available for users to manipulate.

- The standards used for data representation over Internet, and the visualization standards should be designed and developed.

2.3.2 Data Exchange among Applications

As mentioned in Chapter 1, two strategies could be employed for product data exchange in computer readable form [Owen 1993], using proprietary direct translators (Figure 2.3) or a public domain neutral intermediate (database) format (Figure 2.4). In general, where data exchange takes place between n systems, $n \times (n-1)$ translators are required for the first strategy and $2n$ translators for the second; consequently, if more than three systems are involved, the neutral intermediate format approach requires fewer translators. If another CAD system is added to an existing scenario, then $2n$ additional translators are required for the first strategy, but only 2 for the second. For maintaining a large number of translation programs with the direct translator approach, detailed knowledge of the both proprietary formats, between which a program is converting, is needed. Each new release of each system from each vendor requires rewriting many programs.

Figure 2.3 Proprietary direct translators [Owen 1993]

Based on the above analysis, it is evident that although direct translator has the advantage of fast and accurate, it is only suitable for small number of systems and sensitive to the change of version. Therefore, it suffers the problem of scalability. Whereas, the neutral intermediate format strategy is suitable for the information sharing among large number of CAx applications.

Figure 2.4 Neutral intermediate format [Owen 1993]

2.3.3 Research Effort for Multi-application Information Sharing

According to [Loffredo 1998 and Wilson 1990], the neutral intermediate format strategy can be further categorized as four levels as shown in figure 2.5.

Figure 2.5 Levels of multi-application information sharing
[Loffredo 1998 and Wilson 1990]

2.3.3.1 Neutral File – Level 1

Neutral file-based approach is characterized as sharing information in physical file format. It is the simplest way for information sharing among multiple applications and easy to be implemented. Proprietary data translators or public translating service must be provided to translate application proprietary data into a neutral format, or in the reverse way. Zhang, *et al.* [Zhang 2000] presented a system design for an Internet-based data exchange among virtual enterprises. It combines the Internet and STEP technologies, and is supposed to provide more effective and efficient mechanisms to support virtual enterprises. Similar research works using file-based approach were reported in [Zheng 1994, An 1995 and Clark 1995].

2.3.3.2 Standard Data Access Interface (SDAI) – Level 2

With all the features of neutral file-based approach, a data access interface is incorporated in this approach for user management, file indexing and shared access control. Therefore, efficient mechanisms must be provided to manage multiple users, index distributed files. This approach has also been described in STEP standard as SDAI binding [ISO1995a]. This aspect is also reviewed in more detail in section 2.3.3.3.

Chao *et al.* proposed to use an index server for searching distributed files [Chao2001]. The index server can locate distributed files by IP address and communicating with multiple clients. Lee *et al.* [Lee 1999] proposed a network-centric feature-based modeling system. In order to alleviate communication load between server and client, AAB (attributed abstracted B-rep), an abstracted and simplified B-rep, which can realize much data reduction, was described to represent client side feature model. AAB can be realized by providing generic naming schema and mechanism for AAB update.

Li *et al.* [Li 2004] presented a system to support feature-based design in distributed collaborative environment. The face-based data structure in client side can greatly reduce

communication load between server and client. A modeler-supported server provides feature modeling and feature validation functions in the server side. Others solutions based on the same approach have also been developed, such as ST-Developer ROSE, NIST SCL [Clark 1990], SDAI libraries [Hardwick 1991], and [Sauder 1995]. However, as Mittra stated [Mittra 1991], file-based approach has the following disadvantages:

- Redundancy of the data. Identical data are distributed over various files (e.g. different CAX applications share the same final part geometry, but these geometrical data has to be duplicated in different files).
- Multiple updates and potential conflicts. Updating of multiple files consistently is not supported and thereby leading to the lack of data integrity and potential data conflicts.
- Waste of storage space. When the same data is stored in different files, the required storage space is needlessly increased.
- The access language is normally unique to application programs and often not user friendly.

In addition, file-based approach does not provide an integrated view of product information and cannot implement a information model that is generic for different application programs. Last but not least, files are often stored in the distributed clients' sites; it is difficult for other applications to search useful information.

Although data access interface can control shared accesses for multiple users and efficiently index distributed files, it still cannot share fine grain product information in a uniform and consistent manner. It has to combine with a neutral file, hence it still suffers from the problems of file-based approach.

2.3.3.3 Database – Level 3

This approach can be characterized as sharing information through data access interface with database support. It has all the features required by level two, plus the ability to work with data stored in a Database Management System (DBMS). Compared with file-based approach, database approach has the following advantages [Ramakrishnan 2000, Fortier 1997 and Mitra 1991]:

- Naturally shared data for multiple users.
- Reduction or elimination of data redundancy.
- Maintenance of data integrity and security.
- Reduction in wasted storage space.
- Access to data through a query language that describes the information needed by the user but does not need to specify how the information is to be derived.

In addition, databases can manage large amount of information and are very powerful for reading, writing, updating and deletion operations. The DBMS can also ensure the security and transparency for the users of CAD data. Therefore databases are appropriate tools for information sharing among different CAx systems. Ideally, an integrated product database can store data that covers all aspects of the entire product life cycle. Multiple applications can access the product data, and may take advantage of database features such as query processing.

➤ *Geometry Database*

ESPRIT's (European strategic program for research and development in information technology) project 322 "CAD Interface", also referred to as CAD*I, was among the first to use DBMS to realize the data exchange among different CAD systems. In the project, relational databases were used as the physical repository. Geometrical data (B-rep models with planar faces) and some administrative data were stored across different database

tables. It also provides data access interfaces for the management of large amount of information in file or database format. The results of this project showed that with a relational database management system, all kinds of data used for applications and operations can be well managed. The security of the data can be ensured. In addition, the communication and productivity can be greatly improved [Raflik 1990]. However, the project results were not up to the expectations from CAD suppliers and CAD users because of the following drawbacks:

- Geometrical representation schema adopted in this project is in proprietary format which means it is not applicable for other applications.
- CAD*I database covers only part of the geometrical data (B-rep with planar faces) and some administrative data. Therefore, it is not enough to represent different aspects of the entire product life cycle.

Therefore, a lot of work has yet to be done. In order to describe a product's entire life cycle, many STEP-based efforts have been carried out since the approach is most promising. Regli *et al.* described a repository for design, process planning and assembly based on STEP [Regli 1997]. The database repository provides public accessible 2D and 3D CAD and solid models from industry cases. Although using database as the physical repository, models in database are still stored in multiple exchanged file format. Therefore, this approach still suffers from the problems of file-based approach.

Under the STEP framework, a web-based collaborative design system is proposed and implemented as Cyberview by Kang [Kang 1997] and Kim [Kim 1998]. They used VRML as the data communication format over internet and OODB (object-oriented database) as the physical repository to build a collaborative design environment. Domazet *et al.* [Domazet 2000] described an inter-organizational information infrastructure for collaborative product development. It incorporates an Object-oriented database as physical

repository and adopts CORBA and STEP as the open communication and product standards. Loffredo [Loffredo 1998] focused on the efficiency analysis of an engineering database built on the basis of STEP AP 203. El-Mehalawi *et al.* [El-Mehalawi 2003a, 2003b] built a mechanical component database by extracting attributed graph from STEP files and indexing it. In this way, efficient indexing mechanisms based on geometric and topological similarity can be achieved in the proposed component database. However, the above mentioned research works have the following limitations:

- STEP AP 203 contains only geometrical representation schema. This means higher-level feature information (semantic information) is lost. Therefore, it can not support complete information integration.
- In the database design, no mechanisms are provided to control concurrent access to product model data while maintaining data consistency and security.

➤ *Feature-oriented Database*

Currently, most of the CAx applications are parametric and feature-based. Therefore, feature information (feature semantics) should be maintained [Bronsvoort 1997, 1998, 2001; Martino 1998; Bidarra 2001 and Berg 2002]. Urban, *et al.* [Urban 1993] presented a framework that views an engineering design environment as an integrated, heterogeneous database system. This integrated view of data makes use of data exchange standards in PDES/STEP and supports an object-oriented query language for the retrieval of data from multiple data sources. Active database technology is adopted to support a rule-based approach for constraint propagation and change notification. However, features, even those form-features defined in STEP [ISO 2000b] had not been supported. To model a product database for collaborative product development, da Cunha *et al.* [da Cunha 2002] and Lee *et al.* [Lee 2005] proposed to use object-oriented way to define a feature-based product database. However their proposed approach is still at a conceptual level and has not been

detailed for real product design. No detailed database schema and supporting mechanisms were given.

Hoffman *et al.* [Hoffman 1998] proposed the product master model to integrate CAD systems with downstream applications via different feature views throughout the stages of a product life cycle. A product change protocol was proposed to maintain the links between application proprietary data (feature information) and the shared product master model stored in a central database. Theoretically, this approach is feasible. However, the proposed product master model contains only shared geometrical data, namely, the net shape of the product model. To maintain the association between product master model and the distributed application proprietary feature semantics will be very difficult practically. In addition, although those distributed clients in the same domain (e.g. CAD domain) can access the master model concurrently, but the consistency maintenance of the product master model with constant updates remains a problem.

Wang, *et al.* [Wang 2002, 2003] put forward a collaborative feature-based design system to integrate different CAx systems with database support. The definition, classification and relationship of features are described. To alleviate communication load between client and server, Koparanova *et al.* developed an EMQ (engineering mediator query) system that can support collaborative design by sharing information [Koparanova 2002]. Great effort has been taken to validate model geometry. The drawback of the proposed structure is the lack of geometrical engine to support model validation. Recently, the concept of feature operation, was defined by Chen *et al.* [Chen 2005], and realized using KQML (Knowledge Query and Manipulation Language [Finin 1992], [Finin 1993], [Labrou 1997]). However, detailed database design is not given; more research has to be done to investigate how features, constraints and geometrical entities are to be managed.

Furthermore, the mechanism for updating feature model based on operations in the database server side has not been developed.

Kim *et al.* describes an interface (OpenDIS) between the geometric modeling kernel and the DBMS for the implementation of CAD system that uses the STEP database as the native storage [Kim 2003 and Shin 2000]. A prototype system was developed using OpenCascade geometric modeling kernel and ObjectStore. STEP AP 218 was adopted as the information model to build the database. However, since STEP cannot fully cover feature information for different CAx applications, therefore, using only STEP-based product specification is not sufficient to ensure a fully-integrated comprehensive product model. In order to integrate different CAx applications, a generic feature model must be defined such that application-specific feature model can be defined and entire product model can be generated.

Xue *et al.* [Xue 2003, 2004] proposed a distributed and feature-oriented database modeling approach for concurrent design by adopting VRML and Autocad as the standard data format and the geometry modeling kernel among multiple users. Distributed databases are associated by their relations, which are identifiable through the web. However, their distributed databases can not eliminate duplicated data, e.g. geometric data shared by different applications is duplicated in different databases. In addition, it is not clear how concurrency and consistency can be managed, especially for multiple users in the same domain participating in the design of the same part. Furthermore, in such a distributed environment, the coordination among multiple users and databases will be very difficult practically. Similar researches on database approach include [Krebs 1995], [Jasnoch 1996], [Xiao 1996] and [McClatchey 1998].

2.3.3.4 Knowledge Base – Level 4

The database can be further expanded to support the knowledgebase by adding methods for constraint validation and ability to reason about the contents of the database. Knowledgebase systems encode rules by using techniques such as frames, semantic nets, and various logic systems, and then use inference techniques such as forward and backward chaining to reason about the contents of a database [Loffredo 1998]. In this field, preliminary research on knowledgebase was done by PreAmp project [Muller 1993], [Gadient 1994]. An AP 210-based knowledgebase for analyzing the manufacturability of electronic circuit boards is built. The key component in the system, a manufacturability advisor, loads data into an Intellicorp Kappa database, where rules analyze aspects of the design. Later, Wu, *et al.* [Wu 1999] proposed an Internet-based knowledge sharing and collaboration system model of three-layer architecture. This system supports HTTP (Hyper Text Transfer Protocol) requests for information retrieval and browse. The information sharing is in XML (Extensible Markup Language [Bray 1997a, 1997b, 1998]) format. Kim, *et al.* [Kim 2001] explained the design and development of a distributed, open, and intelligent product data management system by incorporating CORBA, STEP as well as KQML. However, the research on knowledgebase implementation is still in a preliminary stage.

2.3.4 Commercial Effort

Except research effort, there also exist commercial solutions which can support collaborative engineering to some extent [Domazet 2000].

2.3.4.1 CAD Web Portals

CAD web portals such as CAD/CAM-E [CAD/CAM-E] and OpenDXM [OPENDXM] are in fact operated by translator providers. They take certain kinds of file format as input and generate another file format, e.g. from IGES to DWG (Drawing). During data

translation, feature information cannot be converted due to the different feature definitions in different CAx applications. Therefore, such information sharing is not complete.

Recently, a commercial system from Proficiency performs feature translations between major CAD systems (Unigraphics, Pro-Engineer and CATIA). The principles behind is on the basis of Rappoport's research work. In [Rappoport 2003 and 2005] and [Steven 2004], a new method, GPF (geometry per feature) has been introduced to realize feature-level translation among commercial CAD systems. UPR (universal product representation) is used as the central representation (a kind of neutral intermediate data format). The implementation, Proficiency's product CG (collaboration gateway), has been successfully used for real product data exchange from single part to huge assemblies.

2.3.4.2 Groupware Tools

AutoEDMS, provided by ACS software, is one of the earliest EDM (engineering data management) system to manage CAD files (AutoCAD, Microstation, CADKey, Computervision, CADAM, etc.) [ACS software] [AutoEDMS 2001]. It has been used by major engineering firms, for departmental and enterprise-wide management of CAD drawings, scanned images, electronic documents, hard-copy files as well as workflow via network (e.g. LAN (local area network), WAN (wide area network) and Internet) with client-server architecture. It can be integrated with other applications such as ERP (Enterprise Resource Planning) or MRP (Manufacturing Resource Planning) systems. However, AutoEDMS is essentially a document base managing the CAD data in file format, which does not solve the problem of data inconsistency. For example, CAD and CAM files for a part should share the same geometric data; such links are not supported by EDM systems. At the same time, files are also not flexible enough to support multi-views required by different applications.

Unigraphics provides an approach called QDM (Quiet Data Management), which is an enterprise-wide PDM solution [PDMI2 1999]. All the CAD/CAM files are stored in the same IMAN PDM vault so that it can be easily accessed. Manufacturing engineers can see the released versions while the design engineer is looking at the engineering change orders (ECOs) and each will see only what is relevant to them. There is no wall between engineering and manufacturing because all the data is in the same system. However, current PDM systems have the following deficiencies that limit their effectiveness for collaborative product development [Domazet 2000]:

- A PDM system is basically still a kind of file-based data management and cannot provide flexible information sharing with greater or finer granularity.
- It is difficult to maintain data consistency and integrity of a product model stored in many files in different locations and various formats.

Another commercial system, OneSpace by Ccreate [Emmel 2000], also offers some collaborative modeling capabilities. However, its underlying techniques are not clear due to commercial protection, e.g. information management at the server side, concurrency and consistency control. Bidarra et al. stated [Bidarra 2001]: "... the modeling facilities of Onespace are severely constrained by the modeler at the server, SolidDesigner, and by the model format into which it converts all shared models".

2.3.5 Integration Platform

2.3.5.1 Computer Network-Based Integration

The concepts of Intranet, Extranet, and Infranet [Pfeifer 1999] have been widely applied. More importantly, Internet, a worldwide system of computer networks, has become an essential broad platform to implement global design and manufacturing. Intranet is a private network that is contained within an enterprise. It channels the

mainstream of enterprise information. Extranet is a private network that uses the Internet protocol and the public telecommunication system to securely share part of a business's information or operations with suppliers, vendors, partners, customers, or other businesses. Extranet is a network formed among enterprises to realize virtual enterprises. Intranet, referring to the infrastructure of a network, is utilized to achieve a blend of control network and data network.

Computer network offers promising features. For example, the client-server architecture provides both the flexibility and the control for the sharing information among global collaborative users across the time and space zones. The multimedia capability of the computer network can also be used to develop network-centric applications that are equivalent to stand-alone systems in terms of functionality, performance and usability [Huang 2001].

One of the first and most significant initiatives in the development and application of web-based systems in product design and manufacture is the US research project- MADE (Manufacturing Automation and Design Engineering) program. Initiated in 1992 and completed in 1996, the MADE program supports research, development and demonstration of the enabling technologies, tools and infrastructure for the next generation of design environments for complex electro-mechanical systems [Bryant 1996, Cutkosky 1996, and Petrie 1996].

Recently, a number of computer network-based systems have been developed to support design collaboration. For example, a web-based tool and requirement management system was reported by [Jiao 1999]; a design project management system by Salas and Rogers [Salas 1997]; and a web-based collaborative feature modeling system by Bidarra [Bidarra 2001].

2.3.5.2 CORBA

CORBA (Common Object Request Broker Architecture) is a standard architecture for distributed object systems [Mowbray 1995]. It allows a distributed, heterogeneous collection of objects to communicate across the network, and has been broadly used to develop collaborative systems. The basic structure of CORBA is illustrated in figure 2.6. CORBA separates interface from implementation and provides language-neutral types that make it possible to call objects across language and operating system boundaries. Using CORBA IDL (Interface Definition Language), it is possible to make existing code look like an object on the ORB. Therefore CORBA can enhance the system integration.

A new information integration platform for computer integrated manufacturing system (CIMS) is presented in [Chan 2000], which is based on agent and CORBA. Joe, *et al.* [Joe 2000] reported a pilot application of a decision-making environment for part manufacturability assessment in extended enterprises using collaborative design and manufacturing agents. The system is implemented in an XML/CORBA based environment for conveying design and manufacture information across traditional technology boundaries.

Figure 2.6 Basic architecture of CORBA

2.3.6 Database Technology

Database is designed to handle a large volume of data, provides generic methods for reading, writing, updating and deleting operations. The database technology can be used to ensure the security and transparency for CAD data with greater flexibility than using physical file format.

2.3.6.1 DataBase Management System (DBMS)

DataBase Management System (DBMS) is a software system that controls shared accesses to a database and provides mechanisms that ensure the security and integrity of the stored data. The main objectives of a DBMS [Ryan 1995] are to:

- Provide the means to define and construct a database.
- Provide efficient and reliable shared access to the stored data.
- Support multiple views of a common data resource to suit the needs of different functional areas and applications.
- Maximize reliability and availability by provision of efficient backup and recovery systems.
- Reduce the need for application maintenance by supporting a high level of independence between applications and the logical and physical structure of the database.
- Provide high-level tools to simplify application development.

2.3.6.2 Types of Database

The commonly used database types are relational, object-oriented and the object-relational.

Relational database management system (RDBMS), based on the relational model proposed by E.F. Cod 1970, organizes data into tables. The rows and columns of a table represent records and attributes. Relational databases use Structured Query Language (SQL) as the data definition language (DDL) and the data manipulation language (DML). The relational model has been adopted in products like ORACLE, DB2, INGRES, INFORMIX, SYBASE, etc. These products are characterized by a better physical/logical independence, greater flexibility, and specification languages.

Object-oriented database management system (OODBMS) is built on the basis of *The Object-Oriented Database System Manifesto* [Atkinson 1989]. This manifesto was published by a group of researchers and has established the principles followed by OODBMS products like O2, ObjectStore, Objectivity, ITASCA, GemStone, SERVIO, etc. An object-oriented database employs a data model that supports object-oriented features such as encapsulation, inheritance and polymorphism. It utilizes the power of object-oriented programming languages to provide excellent database programming capability.

A hybrid of RDBMS and OODBMS appeared recently called object-relational database management system (ORDBMS) [Date 1998 and 2000]. The main objective of ORDBMS design was to benefit both the relational and the object models in scalability and support for rich data types. An ORDBMS employs a data model that attempts to incorporate OO features into RDBMSs. All database information is stored in tables, but some of the tabular entries may have richer data structure, termed abstract data types (ADTs). An ORDBMS supports an extended form of SQL called SQL3 in order to support ADT's. It has the relational model in it because the data is stored in the form of tables having rows and columns and SQL is used as the query language and the result of a query is also table or tuples (rows). Large RDBMS vendors such as ORACLE, IBM, Informix now are evolving toward this direction. Their products are ORACLE 9i, DB2, Universal Server respectively.

There are a lot of arguments for whether RDBMS, OODBMS or RDBMS should be used. Here, we don't intent to be involved in such arguments. The reasons for why we have chosen ORACLE 9i (ORDBMS) are listed below:

- ❑ Data modeling. Although OODBMSs are suitable to handle with complex data models, ORDBMSs, with the help of incorporated OO features and certain mapping mechanism, can also fulfill this task. For detailed mapping mechanisms, please refer to chapter 4.
- ❑ Support large-scale systems. Our research aims to build a web-based data management system to enable data exchange or sharing for concurrent and collaborative engineering. The large amount of product and process information and multiple users require that the DBMS supports large-scale applications. Comparing with OODBMS, ORDBMS has good track records of supporting large-scale systems and providing large storage capacity.
- ❑ Optimization and performance. ORDBMSs also maintain the RDBMS advantage of better optimization and performance than OODBMS because of the good solid foundation of the relational model (algebra, calculus). For OODBMS, optimization is more difficult and complex, because of the lack of algebra.
- ❑ Application development tools. It is essential that DBMSs provide different kinds of development tools (forms and report generators, screen painters, etc.). ORDBMSs, supported by large vendors, are more mature in this sense than OODBMSs.
- ❑ Monitoring and administration tools. ORDBMS vendors offer lots of tools to administrate, tune, and monitor databases. Object-oriented databases are relatively poor in this aspect.

- ❑ Data migration. Another big obstacle to adopt object-oriented database is the lack of an easy and non-disruptive path from relational database to object-oriented database. With the help of standard SQL, ORDBMSs, built on the relational model, do not suffer this problem.

2.3.6.3 Architecture of DBMS

The basic DBMS architecture is the client-server architecture which separates the processes that use and provide a service. Client and server processes may both run on a single computer or on separate computers with communications via the network. There are two types of client-server architecture: two-tier and three-tier architecture.

- ❑ Two-tier architecture

Two-tier architecture as its name suggested consists of only client and server tiers. The client tier manages not only the user interface and displaying the query results, but also the processing logic to manipulate the database. Therefore it is also called fat client. The server is responsible for the data storage. Figure 2.7 illustrates the structure of two-tier architecture.

Figure 2.7 Two-tier database architecture

□ Three-tier architecture

Three-tier architecture consists of client, application server and database server tiers (see figure 2.8). In this architecture, client is typically an application program responsible for displaying query results and managing the user interface. It is called thin client tier. The application server processes the user's requests such as queries and updates, and manages the database. The database server is responsible for the data storage.

Figure 2.8 Three-tier database Architecture

2.3.6.4 Database Schema

Tsichritzis *et al.* [Tsichritzis 1978] proposed the three-level schema database architecture as illustrated in figure 2.9. The three schemas are external, conceptual and internal schemas. Internal schema specifies the physical storage model containing information such as file location, structures, indexing and access methods. External schema is coming from specific application area. A database may contain several external schemas. Each external schema is a subset of the overall conceptual schema. Conceptual schema is the single consistent logical model of the complete database.

Figure 2.9 Three-level schema DBMS architecture [Tsichritzis 1978]

The three-level schema provides DBMS extensibility because in order to integrate an application, you merely need to add an external schema. In addition, the three-level schema makes database natural to support multiple views while at the same time maintaining the data consistency.

2.4 Feature Technology

2.4.1 Feature Definition

Although the existing data exchange standards are the results of great effort for information sharing, however, most of them only focus on lower-level geometrical data. STEP has been extended to cover feature information, but it is limited to form features [ISO 2000b]. Currently, most of the CAx systems are parametric and feature-based. The advantage of using features in product development comes from the abstraction of feature information that retains not only the associative geometric information but also much of the useful non-geometric information that reflects engineering semantics within different applications [Liang 1998]. Therefore, higher-level feature information must be represented

and associated such that engineering meaning is fully maintained and shared among different users.

Features are originated in the reasoning processes and used in various design and manufacturing activities [Cunningham 1988]. They are used to associate functional information with shape information. A feature definition given by [Shah 1995] is that a feature:

- is a physical constituent of a part;
- is mappable to a generic shape;
- has engineering significance; and
- has predictable properties.

In the early stages of feature technology development, features were usually predefined as parametric templates, which have precise geometries for feature-based modeling systems to initiate a feature object or for feature recognizers using pattern matching to recognize an application feature. For example, Joshi *et al.* represented each feature as a fixed attributed adjacency graph (AAG) [Joshi 1988]. In such definitions, only feature syntax, i.e. the topological and geometric relationships between different geometric entities of a feature are precisely predefined and these relationships are usually fixed. These feature definitions have two limitations. Firstly, they lack the flexibility to be extended. Secondly, the lack of specifications of feature semantics may result in information consistency breakdowns [Otto 2001]. For solving these limitations, several approaches are proposed. Bidarra, *et al.* [Bidarra 2000] proposed a semantic feature modeling approach based on cellular topology in 2000. He explained in detail the semantic problem suffered by most current feature modeling system, e.g. ill-defined feature semantics and lack of semantic maintenance.

Another issue about feature definition is the implementation method. There are two approaches, i.e. procedural or declarative ones [Ma 2003]. The major difference of these two approaches is that specifications of feature properties and procedures for manipulations of a feature instance are combined together in a procedural function of software while they are decoupled in declarative feature definitions. Object-oriented (OO) feature representations are typically declarative. A merit of declarative feature definition is that more modularity is achieved by the separation of feature definitions and feature instantiations. Further more, in OO approach, the class definition with properties and methods' protocols are separate from the implementation functions, hence, a unified specific system independent feature implementation is possible [Shah 1995] [Venkataraman 2001].

2.4.2 Feature Information Sharing

Traditionally, higher-level feature information cannot be exchanged among different applications. More recently, researchers such as Bhandarkar *et al.* [Bhandarkar 2000], Dereli *et al.* [Dereli 2002] and Fu *et al.* [Fu 2003] proposed different algorithms to identify useful feature information from the exchanged part models. Although feature extraction [Holland 2002] and identification can partially recognize some feature information from the exchanged part models, information loss still occurs because these approaches depend on pure geometric data. For example, feature relationships (constraints) cannot be recovered from the geometric data model.

In order to enable higher-level feature information sharing among different applications, many researchers [Suh 1997, Martino 1998, Jha 2000, Li 2002 and Gao 2004a, 2004b] proposed to use design information as the input to derive downstream application feature models by feature conversion. However, their works support only one-way link which

means they can only convert from design features to other application features. In [Bronsvoort 1997, 2001, 2004 and Noort 2002], the multi-view feature modeling approach that can support multi-way feature conversion by feature links, is proposed. An “associative feature” definition was developed in [Ma 2003, 2004] for establishing built-in links among related geometric entities of an application-specific and multi-facet feature while self-validation methods were defined for keeping feature validation and consistency. Compared with one-way feature conversion approach, these multi-way feature conversion approaches are promising to support multi-view product modeling.

The concept of unified feature model was first proposed by Geelink *et al.* [Geelink 1995]. The interactive definitions for design and process planning features were focused. However, the constraints defined were limited within one application feature model. Therefore, different application views could not be integrated in this model. Chen *et al.* [Chen 2004a, Chen 2004b] further enhanced the unified feature modeling scheme by introducing inter-application links for higher-level feature information sharing among different CAX applications. Unified feature model is essentially a generic semantic feature model for different CAX applications covering three-level relations among geometric and non-geometric entities. Unified feature model is further extended to knowledge-based model by incorporating knowledge reasoning technique [Chen 2005a, Chen 2005b].

In the above-mentioned feature technology research work, no database support is mentioned which means they probably managed product information in file format. The disadvantages of file-based approach have been addressed in section 2.3.3.1. Comparatively, database, by its nature, support such kind of inter-application links with great flexibility.

2.4.3 Constraint

2.4.3.1 Constraint Classification

One of the main tools on which feature-based modeling relies is constraint modeling [Hoffmann 2002]. Currently, almost all feature-based modeling systems use constraints to specify relationships among geometrical entities, topological entities and features. A *constraint* is defined as a specification of a relation that should be hold and associated to one or more entities or constraint variables [Dohmen 1998].

Constraint may have various types, such as geometric, algebraic, dimensional, semantic constraint as classified by Dohmen. Geometric constraints specify the geometric relations between feature elements. Algebraic constraints define the relations among feature parameters using equations. Dimensional constraints specify distances between two feature entities. Semantic constraints specify topologic properties of feature elements.

Bettig and Shah gave a detailed classification of geometric constraints for the shape definition in CAD [Bettig 2001]. In there classification, the geometric constraints can be categorized as *distance*, *coincident*, *incident*, *tangent*, *offset_curve*, *angle*, *same_direction*, *right_angle*, *symmetric*, *parameter values* and other specific types such as *axis_angle*, *axis_distance*, *fixed_point*. For detail of this classification, please refer to [Bettig 2001].

2.4.3.2 Constraint Solving

Constraint solving mechanisms are widely used in graphical applications [Kwaiter 1998]. A constraint solving mechanism aims at determining whether it is possible to satisfy a set of constraints, and if it is possible, then assign the values to constrained variables. In CAD domain, many constraint solving mechanisms have also been developed. They could be classified according to whether they solve the constraints sequentially or simultaneously [Laakko 1996]:

- When some variables are known, sequential constraint solving mechanisms use local propagation methods to calculate other unknown variables to satisfy the constraint. Local propagation methods are rapid and support incremental modifications of variables, but this method is not possible for solving cyclic constraints.
- Simultaneous constraint solving mechanisms translate a set of constraints into a system of linear or nonlinear equations and solve these constraint equations simultaneously using numerical methods. This kind of method can solve cyclic constraints, but any modification of variables requires the complete re-evaluation of the whole set of constraint equations.

A brief review of major constraint solving mechanisms used in CAD domain is given below:

- Light and Gossard [Light 1982] proposed using Newton-Raphson method to solve a system of linear or nonlinear constraint equations simultaneously. This method only supports functional constraints, requirement constraints and numeric constraints.
- Sannella proposed the SkyBlue algorithm [Sannella 1993a and b], a local propagation method, which supports multi-way constraints, constraint hierarchies, and multiple-domain constraints.
- Borning *et al.* [Borning 1996a, b and 1998] proposed a local propagation method, 'Indigo', for solving inequality linear constraints.
- Thornton *et al.* [Thornton 1994a, b and 1996] proposed a simulated annealing algorithm to solve large sets of cyclic, non-linear, equality and inequality constraints.

- Fudos *et al.* [Fudos 1997] proposed a graph-constructive approach which can handle well-constrained, over-constrained and under-constrained geometric constraints.
- Kumar *et al.* [Kumar 2000] used sequential approach to impose geometrical constraints to positioning rigid bodies during feature creation as well as assembly planning.
- Li *et al.* [Li 2002] used a DOF (degree of freedom)-based constructive approach to solve geometric constraints, which can also handle well-constrained, over-constrained and under-constrained situation on the basis of dependency analysis.
- Lee *et al.* [Lee 2003] proposed a graph constructive approach which can handle ruler-and-compass non-constructible configurations, and under-constrained problems.

2.4.4 Geometrical Modeling Kernel

2.4.4.1 Overview

All feature-based modeling applications have to be built around a geometry modeling kernel, which is also referred to as the modeling engine or the geometry modeler. A geometry modeler can provide lower-level geometrical modeling services to support higher-level parametric feature-based modeling.

Currently, there are only a few geometrical modeling kernels are commercially available. More dominantly, ‘Parasolid’ by UGS Inc. and ‘ACIS’ 3D by Spatial Inc. are two well-known geometrical modeling kernels that were adopted by most commercial CAD systems. Both ACIS and Parasolid have a similar set of functionalities. Historically, ACIS had come from a more linear solid background that made it a more natural fit for CAM while Parasolid had excelled more in blending and curving, which made it better in

consumer products. Both kernels are evolving and any comparison will become outdated in a matter of months if their development schedules are adhered to [Green 1997]. There is only one thing different: compared with Parasolid, ACIS geometrical modeler is free for research purpose. Therefore, ACIS has been adopted in this research.

2.4.4.2 ACIS 3D Geometrical Modeler (Spatial/Dassault systems)

ACIS 3D geometrical modeler is an object-oriented three-dimensional (3D) geometric modeling engine from Spatial [ACIS 2001]. It is designed as the geometry foundation within virtually any end user 3D modeling application. ACIS is written in C++ and consists of a set of C++ classes and functions. ACIS complements existing applications by offering a unified environment for the modeling of curves, surfaces, and solids. It also supports the integration of proprietary curve and surface subsystems. ACIS provides a foundation of common modeling functionality and the flexibility to be adapted and extended for particular application requirements. ACIS integrates wireframe, surface, and solid modeling by allowing these alternative representations to coexist naturally in a unified data structure. ACIS bodies can have any of these forms or combinations of them. Linear and quadric geometry is represented analytically, and non-uniform rational B-splines (NURBS) represent free-form geometry. In addition two-manifold geometry, ACIS can represent non-manifold geometry. Geometry can be bounded, unbounded, or semi-bounded, allowing for complete and incomplete bodies. A solid can have faces missing and existing faces can have missing edges. Solids can also have internal faces that divide the solid into cells. ACIS is a dual-format solid modeler with combined or separate boundary-representation (B-rep) as well as cellular model representation. This means that it defines the boundary between solid material and empty space in addition to decomposed cells. This boundary is made from a closed set of surfaces.

2.5 Technological gaps

Ideally, in order to support collaborative product development, an information sharing system should:

- Use international standard, since such standard allows participation by a wide variety of parties, including suppliers and customers.
- Provide unified product information model to cover entire product life cycle such that different applications can be integrated. Further more, it should contain not only the geometry, but also higher-level feature information so that engineering meaning can be maintained.
- Provide physical repository and efficient mechanism to manage large amount of product data.
- Manage the information in a concise and effective manner such that the entire product lifecycle information needs to be generated only once. Duplicated data is avoided.

As reviewed in this chapter, although a lot of research and development works has been done on information exchange/sharing to enable concurrent and collaborative engineering, technological gaps between the industry requirements and the available solutions still exist as follows:

- *Duplicated data and conflicts.*

Product and process information is often stored in a file format [An 1995, Clark 1995, Zhang 1999, 2000 and Chao 2001], which means duplicated data and potential conflicts. In addition, files are not flexible enough to support the multiview functions required by concurrent and collaborative applications. Furthermore, it is difficult to extract and manage useful information from distributed files. For example, multiple end-users or applications

cannot synchronize definitions and modifications easily for the data stored in individual files.

- *Information loss.*

As for geometrical information, although many existing systems claim to support CAD-neutral (e.g. STEP-based) data exchange, they lack feature level interoperability so far [Regli 1997, Kang 1997, Kim 1998, Loffredo 1998 and Domazet 2000]. Useful information such as features is often stripped off during data exporting and importing. Therefore complete information sharing has not been achieved.

- *The lack of a generic feature definition.*

Most of the database design lack of a higher-level generic feature definition, which can be used as a template for database schema extension in the future. Therefore, these proposed databases suffer from the scalability problem.

- *The absence of efficient feature-oriented engineering database.*

Although there are a few research works as reviewed in section 2.3.3.3, which have focused on building an engineering database to support collaborative product development, many of them focus on higher-level conceptualization [Domazet 2000, da Cunha 2002, Wang 2002, 2003 and Lee 2005] without explicit database schema definitions. In addition, most of them did not provide efficient and effective mechanisms to support product feature information management in a consistent and concurrent manner. Last but not least, without higher-level information integration infrastructure, their works suffer from the problem of extensibility.

The following aspects are identified for investigation in this research:

- An integrated product information model, which includes geometrical model and a generic feature model, to be built on the basis of STEP;

- Mapping mechanisms from EXPRESS-defined model to ACIS workform format and the target database schema;
- A feature-oriented database design with effective and efficient methods to enable information sharing in consistent and concurrent manner;
- The system architecture to enable information sharing among different applications in collaborative engineering environment.

Information integration is very important for collaborative product development. A common information infrastructure that can support information sharing among a large number of CAx systems is urgently required. Among all the existing standards for the data exchange, STEP, which is an international standard, is more promising. However, it is appreciated that STEP feature definition needs to be further investigated for flexible and semantic feature information sharing. Unified feature modeling approach [Chen 2003, 2004a] enables higher-level feature information sharing among different CAx applications. It provides a generic semantic feature model for different CAx applications covering three-level relations among geometric and non-geometric entities. Therefore, this work will be carried out under unified feature modeling approach. Compared with physical file-based approach, database is more generic, flexible, transparent and security-enhanced. This technology is suitable for global engineering information sharing across a virtual enterprise.

Based on the review so far, the research work of this project can be justified - that is to create a feature-oriented database to support concurrent and collaborative product development.

2.6 Summary

A lot of research and development effort has been contributed to information exchange/sharing to enable concurrent and collaborative engineering. As a result, various methods and technologies have been established in the literature for this purpose. This chapter presents a comprehensive review on these technologies. More specifically, it outlines the fundamentals of product design and development, key techniques for information sharing, which includes strategy for information sharing, the existing standard, feature technology, integration platform, database technology and geometrical modeling kernel. In addition, research effort and commercial effort are reviewed. From the review and discussions presented in section 2.5, technological gaps, existing in the field of information sharing for collaborative product development, have been identified. Reducing these gaps is the motivation for the author to conduct this research work.

CHAPTER 3 INFORMATION INTEGRATION

INFRASTRUCTURE AND GENERIC FEATURE MODEL

3.1 Introduction

To achieve information integration among CAx applications, a shared common product model is crucial. It should provide different views for various applications. These views are context-dependent and self-contained interpretations (subsets) of the entire product model (EPM).

In this chapter, a four-layer information integration infrastructure is first proposed for building the shared product model. It is built on the basis of STEP framework because STEP is the international standard and has been widely accepted by both vendors and users. However, using only STEP-based product specification cannot ensure feature model integration, because inter-relationships and constraints between applications are not defined in STEP. In this product model, the STEP framework is extended with a new concept, unified feature model, under which, a generic feature representation schema is given. Next, design and manufacturing feature models are described based on the new concept. The different definitions of *slot* features in both applications are analyzed as examples. For feature-based modeling processes, the concept of *operation* is introduced followed by its representation schema. Finally, the cellular geometric model, shared by different applications, is described.

3.2 Four-layer Information Integration Infrastructure

We propose a four-layer information infrastructure model [Tang 2004] as showed in Figure 3.1 with reference to Zha's work [Zha 2002]. The four layers are application, information, representation and physical layers.

Figure 3.1 Four-layer information infrastructure

Different application models, e.g. design feature model, manufacturing feature model, are represented as sub-models in the application layer; they need to be integrated with application-specific requirements.

The information layer contains four components, i.e. EXPRESS specification, application features, unified features and EPM. The EPM describes information across applications, and contains the domain classification ontology and metadata. This layer contains assembly-part models, product geometry and topology, and the related attributes, etc. In this research, we focus only on the feature-based design and manufacturing model of a product, which includes geometry, constraints, parameters and dimension. Other related information, such as product-related documents and categories, will not be discussed here. A partial schema-level EPM representation is defined as shown in figure 3.2.

Figure 3.2 Partial schema-level EPM (Enhanced from [Tang 2004])

Next, the unified feature model, by giving a generic feature template, provides a generic feature modeling framework for different application feature models. It has been recognized that although different applications define features in different ways, but their features have the common types of data entities, such as geometry, topology, dimensions, tolerances, constraints and parameters. In addition, different application features refer to the same master product geometry. Therefore, the research team has proposed a unified feature model [Chen 2004] as shown in figure 3.3. The unified feature model uses a five-layer architecture. The top layer is the application model layer, which consists of different application feature models. The second layer is unified feature layer. A unified feature has the geometry, the topology, dimensions, tolerances, constraints and parameters as its attributes, which form the third layer. The bottom layer is the geometrical representation

layer. An intermediate geometry_reference layer is used for connecting higher-level feature specifications with lower-level topological entities to solve persistent naming problem. A unified feature model provides generic feature properties and methods to deal with EPM constraints, parameters, dimensions, tolerances, etc; it also associates and interfaces with the EPM geometry, topology, and all kinds of attributes via the geometry reference manager. This manager is used for connecting higher-level feature specifications with lower-level EPM entities to solve persistent naming problem. The unified feature model also contains the domain classification ontology and metadata. It manages product-, assembly- and part-related information and inter-application feature models respectively.

Figure 3.3 Unified feature model [Chen 2004]

Then the detailed high level feature objects are organized by different sub-models in the application feature layer. Application feature sub-models define specific views of the EPM for different applications. All the EPM, application feature model and unified feature model are described in EXPRESS language. For implementation, this EXPRESS-defined information model must be mapped to database schema (for data storage), programming language (workform format) and neutral format (for data communication). Database in the physical layer, provides repository for all kinds of information, e.g. geometrical data,

feature information and others. In the database, all kinds of information are stored as data elements across database tables. Such a kind of data structure makes the information sharing natural, flexible and with levels of granularity. All EXPRESS-G representations in this thesis follow the same convention as shown in figure 3.4.

Figure 3.4 Symbol convention of EXPRESS-G

Note that for page reference, the first # represents page number where the entity is defined, the second # represents reference number of the entity on that page. In this thesis, we omit all the real numbers because we don't want to list all the entities that have been defined by other international standard. Entities with #, # represent entities that have been defined.

3.3 Generic Feature Model

Unified feature model allows different applications to define specific features in a unified approach. Applications features are modeled as the child class of the generic feature. In other words, it allows coexisting specific views for different applications. However, although from application point of view, it is essential that each feature type has a well-defined meaning, or semantics, as a base class, a generic feature definition that

enables common mechanisms, such as data storage, searching, validation, updating and information sharing, is a must to be modeled and developed.

The generic feature class includes the structured description of all common properties and methods of application feature types. Such properties include feature shape representation with parameters, constraint types, reference mechanisms and validity methods. For example, all types of constraints are used for capturing design intent in the context of product design models. The generic feature representation schema for database design and implementation can be represented in EXPRESS-G in figure 3.5 [Tang 2004]. This generic feature model provides a template for application-specific feature definition.

Figure 3.5 Generic feature representation schema

3.3.1 Feature Shape Representation

To represent the shape of a feature means defining feature geometry, topology, their associated entities, such as attributes, parameters, and feature manipulation (e.g. creation, modification and deletion) functions. Feature parameters support user interfaces to create and modify features in the modeling operations. To explicitly maintain the shape of feature in the part model, the shape representation in this research is based on the cellular topology of ACIS, which is on the top of common B-rep model. For example, a *block* feature may

have four parameters as *length*, *width*, *height* and *position point*, see figure 3.6. Creation of a *block* feature is associated with function *api_solid_block()* (creating primitive solid block with two positions). With the above four parameters and feature creation schema, the shape of *block* feature can be determined.

Note that:

- Length parameter is along the X-axis;
- Width parameter is along Y-axis;
- And height is along Z-axis.

Other primitive features with parameters, such as *cone*, *cylinder*, *wedge* and *sphere*, are shown in figure 3.7.

Figure 3.6 Block feature

Figure 3.7 Other primitive features with parameters

Table 3.1 lists other types of features with their basic parameters. Note that in our opinion, datums (which includes datum plane, datum axis and datum point are also regarded as a kind of feature).

Feature type	Primary feature parameters
Slot	1. A 2D profile (e.g. U profile); 2. A path; 3. slot end type (if it's not through slot)
Hole	Type of hole: <ul style="list-style-type: none">● Simple hole→ 1. Radius; 2. Depth.● Countable hole→ 1. Countable_hole radius; 2. Countable_hole depth; 3. Hole radius; 4. Hole depth.
Pocket	1. The pocket profile (rectangular or circular); 2. Depth; 3. Corner radius; 4. Floor radius.
Extrusion	1. A 2D sketch profile; 2. An extrusion path (or direction with distance).
Revolution	1. A 2D sketch profile; 2. An axis; 3. Revolution angle.
Sweep	1. A 2D sketch profile; 2. A sweep guide (path).
Chamfer	1. An edge (chain of edges) or connected faces; 2. Two distances (or a distance with angle).
Fillet	1. An edge (chain of edges) or connected faces; 2. Radius.
Array	Type of array: <ul style="list-style-type: none">● Rectangular→ 1. Arrayed objects; 2. Column offset with number of instances; 3. Row offset with number of instances.● Polar→ 1. Arrayed objects; 2. Axis of polar; 3. Number of instances (with fulfilled angle)
Offset	1. Offset objects; 2. Offset distance with direction.
Mirror	1. Mirror objects; 2. Mirror plane.
Datum	<ul style="list-style-type: none">● Datum plane→ 1. A point; 2. A plane normal(vector);● Datum axis→ 1. A point; 2. A direction (vector);● Datum point→ A point.

Table 3.1 Other features with parameters

3.3.2 Validity Condition (Constraint) Definition

Validity conditions, namely constraints, must be explicitly defined in the unified feature model to specify relationships among features, geometric or topological entities and provide invariant characteristics in the model.

Constraint may have various types. Some classifications, such as Dohmen in [Dohmen 1998], Bettig and Shah in [Bettig 2001], have been given as reviewed in literature review. In this work, we follow the classification by Dohmen (also the same as most current CAD systems). Constraints can be classified as geometric constraints, dimension constraints, algebraic constraints and semantic constraints. The constraint schema has been defined as shown in figure 3.8. Geometric constraints specify the geometric relations between feature elements. For example, in figure 3.9, the start face of the *through_slot* feature coplanes with the front face of the *base_block*. Algebraic constraints define the relations among feature parameters using equations. For example, in figure 5.16 the radius of the *through_hole* equals to half of the radius of *cylinder_boss*. Dimensional constraints specify distances between two feature entities. Semantic constraints specify topologic properties of feature elements. For a vertex, edge or face, a semantic constraint specifies the extent to which the element must lie on the product boundary. For a volume, a semantic constraint specifies the extent to which the volume is allowed to be intersected by other feature volumes instantiated later [Dohmen 1998]. For semantic constraints, an example is shown in figure 3.9. A *through_hole* feature has semantic constraint that the cylindrical side face must at least be partly on the boundary, and the top and bottom face must not be on the boundary. If in the later design stage, a boss is placed just over the hole, the semantic constraint on the top face is violated, as shown in figure 3.9 (b).

Figure 3.8 Constraint representation schema

Figure 3.9 (a)

Figure 3.9 (b)

Figure 3.9 An example for semantic constraints

In the generic feature definition, constraints are modeled as attributes attached to topological entities or sub-features with the associative validation methods defined in its type definition. Although different types of constraints have different attributes, some attributes are common as described below:

- ❖ *Constraint_ID*: It is the identifier of a constraint instance.
- ❖ *Constraint_name*: It specifies the name of a constraint instance.
- ❖ *Owner_ID*: It uniquely identifies which feature a constraint belongs to.
- ❖ *Constraint_expression*: It represents the relationship between the constrained elements and reference elements.
- ❖ *Constrained_entity_ID*: It is used to specify a list of constrained entities with reference to the referenced entities.
- ❖ *Referenced_entity_ID list*: It can be used to uniquely identify referenced entities. In modern CAD systems, the *reference_entity*, which is the existing geometry (a face, an edge or a vertex), is regarded as a kind of datum for positioning (or orienting) a new feature.
- ❖ *Constraint_strength*: It has an enumeration data type, which may include several levels, such as *required*, *strong*, *medium* or *weak*. It represents the extent that the constraint needs to be imposed when constraints conflict with each other. The definition of *constraint_strength* is for handling over-constrained situation. Such design is from an external constraint solver, SkyBlue [Sannella 1993a, 1993b]. The use of *constraint_strength* is supported by this solver. SkyBlue constraints each have a priority, or strength, associated, indicating how important it is to satisfy the constraint. A constraint is said to be weaker than a constraint of a higher priority, which is called stronger. The highest strength is 'required'; the lowest is 'weak'. An arbitrary number of strength levels may be defined. If the SkyBlue constraint graph has conflicting constraints, SkyBlue will always determine a solution such that no unsatisfied constraint could be satisfied by making a weaker constraint unsatisfied. An example is shown in figure 3.10. In the first graph, the strong constraint C2 has just been added. The second method graph shows a possible

method graph without conflicts; in order to satisfy C2, weak constraint C1 is left unsatisfied. Another solution would be to leave C4 unsatisfied instead of C1. Since C1 and C4 have equal strength, SkyBlue will arbitrarily choose one of these solutions. The third method graph shows a solution when the strength of constraint has been set to medium. Now, C4 is left unsatisfied. Note that in the figure, a box represents a constraint. A circle represents a variable or parameter. For details of constraint solving in SkyBlue, please refer to [Sannella 1993a, 1993b].

Figure 3.10 Constraint_strength for constraint solving

❖ *Constraint_sense*: It is used to specify the direction between constrained entities and referenced entities. It has the select data type which maybe *directed* and *undirected*. A constraint is directed if all members of a set or list of constrained entities are constrained with respect to one or more referenced entities. A constraint is undirected if there is no referenced entities and the constraint is required to hold between all possible pairs of a set of constrained entities. Stated differently, in the undirected constraint, there is no difference between constrained entities and referenced entities. For example, if a directed constraint is applied to two lines (line_1 and line_2) and it requires line_2 to be parallel with line_1, it implies that line_1 existed in the model before line_2 was created. The corresponding

undirected constraint would simply assert that *line_1* and *line_2* are parallel, with no implied precedence in their order of creation.

- ❖ *Constraint solving functions*: They are responsible for solving constraint according to constraint types.
- ❖ *Other manipulation functions*: These functions may include attributes access functions, behavior control functions and so on.

3.3.3 Other Generic Feature Properties

Other properties defined in the generic feature schema can be defined as follows:

- *General feature attributes* - *Feature_name* and *feature_id*. General feature attributes such as *feature_name* and *feature_id* are defined to serve as the index for searching of feature during feature modelling operations.
- *Feature type*. Feature type is essentially determined by the instance feature class name derived from generic feature class, e.g. *block* feature, *slot* feature.
- *Depended_feature_id list*. To maintain feature relationship, feature dependency relations should be kept during modeling procedure [Sheu 1993]. Feature dependency relation is described by Bidarra in 1997: “*feature f1 directly depends on feature f2 whenever f1 is attached, positioned or, in some other way, constrained relative to f2*”. Feature dependency relations can be expressed in the feature dependency graph, which is a kind of simplified constraint graph. It is a kind of directed graph, namely digraph (directed graph is a graph each of whose edges is directed). The direction of each edge in the feature dependency graph is from a feature to the feature that depends on it. For example, the part in figure 3.11 can be expressed in constraint graph and feature dependency graph as shown in figure 3.12. Here in the graph, the slot and the two holes are *depended_feature* of the base block feature. *Depended_feature_id* records the feature dependency

relation. It plays important role in maintaining feature dependency graph, and furthermore, maintaining feature relationship during feature modeling operations. Modern CAD systems also keep feature dependency relations. For example, in UG NX 4.0, you can query all such information.

Figure 3.11 Example part

(a) Constraint graph

(b) Feature dependency graph

Figure 3.12 Constraint graph and Feature dependency graph of example part

- *Feature label.* It is an *entity_list* in the feature definition, and used to record feature elements. Feature labels are attached as attributes to feature member entities, e.g. faces, edges and vertices.
- *Domain specification.* As the proposed feature model must support collaborative feature-based modeling among multiple applications, *domain* is used to address which application a feature belongs to. *Domain* has the ENUMERATION data type; its value could be ‘design’, ‘manufacturing’, ‘assembly’ and others. By specifying different domain, multiple application features, which refer to the same product geometry, can co-exist in the feature-oriented database.
- *Nature.* The nature of feature has enumeration data type which is additive and subtractive. Additive means the feature is to be instantiated by adding material. Subtractive means the feature must be obtained by subtracting material.

3.4 Member Functions

- ❖ *Attributes access functions.* They shall be defined to manage feature’s attributes. Most of such functions are common to all type of features, e.g. *backup()*, *findOwner()*, *findConstraint()*, *getParameter()*, *setParameter()* and so on. Other specific attribute methods for individual application features are to be addressed in the application level.

- ❖ *Modeling operation functions.* These functions are used to control the behavior of a feature during a modeling operation, e.g. creating, editing, deleting, splitting owners, merging owners, or translation, etc.
- ❖ *Feature validation functions.* Whenever a feature operation is activated via the user interface, the product model needs to be modified and updated. This process requires feature evaluation, which ensures the consistency of the geometrical model at low levels. In this work, the run-time product model is generated via an integrated solid modeler and managed based on the database records. All feature evaluation functions triggered by such operations call the solid modeler's APIs to realize the geometrical procedures; they are implemented separately in the feature classes. In this way, the details of geometrical operations are readily looked after by the solid modeler; hence, the development effort is significantly reduced. Theoretically, feature process functions can be classified into two kinds, those dealing with the geometry and those managing constraints. With the incorporation of a solid modeler, feature process functions rely on the solid modeler for manipulating and validating feature geometry. Constraint solving functions need to call specific algorithms defined in the individual constraint sub-classes to solve different associative relations according to their types.
- ❖ *Save and restore function.* For repository purpose, feature saving and restoring functions, which are the interactions between the run-time feature model and the database, must be defined in the unified feature model classes because these functions have to organize information for different application views according to users' requirements.

In this research, only design and manufacturing features are considered. In the following section, the design and manufacturing feature models will be described.

3.4 Application-specific Feature Models

A lot of research works have been done for feature classification. Shah and Rogers classify features according to three basic forms: form features, precision features and material features [Shah 1988]. They consider that there are other logical information sets, such as assembly features, functional features, etc. Rossignac clarified distinction between intentional features and their geometric embodiment, and between volume features and surface features [Rossignac 1990]. Juri further classified the form features into primary and secondary, and also into external and internal. The primary features correspond to cylindrical and conical shafts, whereas the secondary are holes, threads, fillets, etc [Juri 1990]. In this work, we only consider design and manufacturing features. Design feature classification used here are similar to those in commercial CAD systems (such as Pro-E, Unigraphics and others). The classification of manufacturing features is based on the AP 224 of STEP.

3.4.1 Design Feature Representation

3.4.1.1 Design Feature Representation Schema

In this section, design features are considered to illustrate how application-specific feature models can be defined. Design feature model can be expressed as shown in figure 3.13.

The primitive feature type is separated into two subtypes, additive and subtractive features. Additive feature covers all instance features formed by adding material such as *cylinder*, *taper*, *sphere*, *boss*, *block*, *torus* and so on. Subtractive feature type represents all features such as *hole*, *pocket*, and *slot* that are formed by subtracting material. The transition feature type includes *chamfer*, *edge_round* and *fillet*, which are always associated to other primitive features. The compound feature type is a union of several

primitive features. Datum, which is used as the reference for feature-based modeling, is also regarded as a kind of design feature in our opinion. Datum has three subtypes, namely datum plane, datum axis and datum point. Other feature types such as extrusion, revolution, sweep and others are also accommodated in this schema. For each specific design feature type, it has predefined explicit geometry, topology, parameterization and constraints specifications. Note that, not all the feature types are included in this schema, because the number of feature type is not finite [Shah 1991]. But by using the generic feature model, feature definitions are extensible.

Figure 3.13 Design feature representation schema

3.4.1.2 An Example Design Feature Definition- Slot

Based on the generic feature definition, a design feature *slot* can be defined in EXPRESS-G as shown in figure 3.14 according to STEP AP 224 [ISO2000b]. *Slot* feature class inherits all the common properties and methods from generic feature class.

- Generic shape representation of *slot* feature

The shape of the *slot* feature is expressed as swept depression volume with a cross section profile and a continuous curve of travel. The parameters associated with a *slot* feature are listed below:

- *Course_of_travel*. The *course_of_travel* specifies a 3D space curve (e.g. line and circle), that when combine with a “profile”, creates the shape of the *slot*. The *course_of_travel* is represented as a *path* in EXPRESS-G in figure 3.15. A *path* shall be defined as the geometrical entity pointer which servers as the input parameter for *slot* feature creation function.

Figure 3.14 Slot feature definition in EXPRESS-G [ISO2000b]

Figure 3.15 Path definition in EXPRESS-G [ISO2000b]

- *End_conditions*. The *end_conditions* specifies the type of implicit shape at the ends of the *slot*, which includes *blind_slot_end_type* or *open_slot_end_type*, as shown in figure 3.16. Different *slot* end types require different parameters. These parameters are associated with *create_slot_end* function that will be called in *slot* feature creation function.

Figure 3.16 Slot end type with parameters [ISO2000b]

- *Sweep_shape*. The *sweep_shape* defines the sectional 2D sweeping profile. When combined with the course of travel, the sweep operation creates the shape of a *Slot*.

For slot feature, the *sweep_shape* is represented as an *open_profile* that could be *square_u_profile*, *rounded_u_profile*, *linear_profile*, *vee_profile*, *partial_circular_profile* or *tee_profile*. To simplify the explanation, only *square_U_profile* is illustrated in figure 3.17. Each type of 2D profile has its own initializing parameters (e.g. *square_U_profile* has two parameters as shown in figure 3.18) which will be used in *create_profile()* functions to create 2D profile. This 2D profile shall be defined as entity pointer and will serve as an input parameter for *slot* feature creation function.

Figure 3.17 Open_profile definition in EXPRESS-G [ISO2000b]

Figure 3.18 Square_U profile with parameters

- Constraints

In the *slot* feature definition, constraints are regarded as attribute list attached to *slot* feature and therefore shall be defined as attribute list. Different type of constraints (e.g. *distance*, *angle* constraints and so on) shall be defined first. All the constraints are treated as common attributes in attribute list of the feature and accessible for the validity check to maintain feature validity.

- Other feature properties
 - Define *feature name* as *slot*,
 - Define *depended_feature_id* as *entity_list*,
 - Define *domain* as 'design',
 - Define *nature* as 'negative'

- Instantiate a *slot* feature

Figure 3.19 An open ended 'slot' design feature with the square U profile

Given the *slot* feature definition described above, instantiating a design feature *slot* shall be carried out in two steps, namely defining the shape of the *slot* as well as positioning the

slot feature. Details are shown as follows (here a *through_slot* feature with *square_U_profile* and a straight line path shown in figure 3.19 is taken as example):

1. Specify the type of the *slot* feature. This is to define the sweep shape (such as *square_U_profile*, *T_profile*, *round_U_profile* or others) of the *slot* feature by specifying the parameters described in figure 3.18. In the instance of *slot* feature, a *square_U_profile* entity with parameters is created.
2. Specify the *slot* end type. A *slot* feature may have end types as shown in figure 3.16. The shape of the *slot* end will be created and combined with the main body of the *slot* feature to get the shape of the *slot*. *Slot* end type with parameters will be recorded as the entity pointer. In the example, for *open_slot_end_type*, no action will be taken for the creating of *slot* end shape.
3. Define a course of travel for the *slot* feature; it can be a line, a circle or 3D space curve. By default, a straight line perpendicular to the sweeping profile will be taken as the course of travel. This course of travel will be stored as an entity pointer. In the example, a course of travel with direction (0, -1, 0) is created by specifying the start and end face of the *slot*.
4. Create the body of *slot* feature by sweeping the profile along the path. This kind of operation will result in a solid as the main body of *slot* feature. In the example, the given *square_U_profile* is swept along the direction (0, -1, 0) for a distance equal to distance between start face and end face of the *slot*. The result is shown in figure 3.19.
5. Position the *slot* feature. To position the slot feature on a planar surface, dimension constraints, which are used to dimension *constrained_entity* (geometry of feature to be created) and *referenced_entity* (existing geometry on the model), are used. In this case, for the definition of *through_slot* feature, two coplanar constraints (C_1

and C_2 are defined to determine the start and end of *slot* feature. Therefore, there is no need to set such a constraint along Y axis. In addition, a distance constraint D, is used to dimension the distance between the *slot_left* face (constrained_entity) and *block_left* face (reference_entity or datum).

- 6. Generate the 3D cell to the shape of the feature on the basis of cellular topology and insert the shape into the part by carrying out non-regular Boolean operation. Details for cellular topology will be described in the later section of this chapter. In the example, the *slot* shape will be Boolean Union with the *base_block* and result in the final part as shown in figure 3.19.

The *slot* feature instance in the example can therefore be expressed as shown in figure 3.20.

Figure 3.20 A design feature (slot) instance

Note that upon cellular decomposition, there are two cells (3D cell) in the cellular model of the example part shown in figure 3.19. The shaded cell represents cell of *slot* shape while the remaining represents the cell of *base_block*.

3.4.2 Manufacturing Feature Representation

3.4.2.1 Manufacturing Feature Representation Schema

A manufacturing feature represents a geometrical shape that is associated with a manufacturing process to produce the associated part faces as designed. STEP AP 224 [ISO2000b] has categorized manufacturing features into three groups, namely, machining features, replicate features, and transition features (see figure 3.21). In this work, due to the limitation of the research scope, only traditional machining features are considered.

Figure 3.21 Manufacturing feature representation schema [ISO2000b]

A machining feature is a subtype of manufacturing feature that is formed by removing solid materials from the initial stock in order to obtain the target part geometry. According to STEP [ISO 2000b] machining features could have the following subtypes: knurl,

multi_axis_feature, thread, marking, spherical_cap, outer_round, revolved_feature and compound_feature. For details of the definition of above mentioned feature types, please refer to [ISO 2000b].

Each machining feature requires direction and position to place it on a part. The data associated with a machining feature are the following:

- Placement

The placement specifies the position and orientation of a machining feature relative to the base shape of a part.

- Usage_name

The usage_name specifies a user defined name that is additional information about the use of a feature. The usage_name need not be specified for a particular machining feature.

A *Compound_feature* is a type of machining feature (see figure 3.22 for example) that is a union of one or more machining feature objects to create a more complex feature definition. The placement of a *compound_feature* is relative to either the part, another *compound_feature*, or a *replicate_feature* which uses a *compound_feature* as the base feature. Features which are elements of the *compound_feature* have placement defined relative to the *compound_feature* placement.

Figure 3.22 An Example of Compound_feature

A *multi_axis_feature* (see figure 3.23) is a type of machining feature that identifies milling features for a part, and not turned features. *Multi_axis_features* have the following types: *boss*, *general_removal_volume*, *hole*, *rounded_end*, *planar_face*, *pocket*,

profile_feature, protrusion, rib_top, slot, and step. The data associated with multi_axis_feature are the following:

- Maximum_feature_limit

The maximum_feature_limit specifies a planar limitation for a feature. The feature can not exceed this planar plane. The normal to the plane shall be in the direction away from the machining feature.

Figure 3.23 Multi_axis_feature representation [ISO2000b]

In our view, unlike design features, manufacturing features depend on the process plan although manufacturing features can have predefined geometry. This fact means the geometry of a manufacturing feature can be determined only after a manufacturing operation has been determined by the process planner. In order to generate manufacturing feature model from the part model, predefined generic feature template library can be used for feature recognition. This procedure can be implemented either automatically or interactively, or in a combined manner. After feature recognition and choosing appropriate machining operations, the shape of manufacturing feature can be determined. Each

manufacturing feature has an associated *machining_operation*, defined and stored as the attributes of the relevant geometrical entities. They can be expressed as shown in figure 3.24. Candidate machining operations are those combinations of the machine tool and the cutting tool, whose capabilities (shape, size, tolerance, surface finish) and accessibility satisfy the required manufacturing specifications. Therefore, the machining operation schema has four major entities: machining method, machine tool, cutting tool and machining operation. The shape of a machining feature contains two volumes, namely, accessing volume and removal volume [Gupta 1994]. In figure 3.25, two machining features, a *slot* and a *hole*, are taken as examples to illustrate their shape representation.

Figure 3.24 Machining_operation schema

Figure 3.25 Accessing and removal volumes of machining features [Gupta 1994]

In a traditional machining operation, material is removed by a moving cutting tool, which is attached to the machine tool. The moving cutting tool together with its chuck, which is driven by the machine tool, will sweep a volume in space (see figure 3.25). The cutting portion of this swept volume is known as the removal volume, i.e. the effective removal volume. The rest of the swept volume is referred to as the accessing volume.

3.4.2.2 An Example Machining Feature Definition- Slot

A machining feature *slot* can be represented in schema format as shown in figure 3.26.

- Generic shape of the machining feature- *slot*

The machining feature *slot* can be used to achieve many kinds of design feature. When a design feature is achieved by applying a slot manufacturing feature, they become associated; but note that they are neither identical nor overlapping. One major difference is that in manufacturing domain, accessing volumes and additional volume should also be considered for manufacturability analysis. These volumes are evaluated when a machining operation is decided. The other difference is the technical details matching between the removal volume and the design feature. The design feature represents the design

requirements, while the removal volume represents machining results. The relations are to be further explored in the future work.

- Validity condition definition
 - The surface type of the *slot* feature must match to the surface type of at least one machining operation in a given manufacturing environment.
 - Tolerance and surface finish specifications of the *slot* feature surface must match to the tolerance/surface finish capability of at least one (usually a set of) machining operations in a given manufacturing environment. It should be noted here that it is usually an ordered set of machining operations, whose total effects equal to or better than the tolerance/surface finish specifications of the finished surface.
 - The effective removal volume of *slot* machining feature cannot intersect with the final design part volume.
 - The accessing volume of the slot shape must not intersect with the blank or work piece, fixtures and other machine tool elements at anytime.
 - When machining the *slot* with an end-milling tool, the minimum corner radius of the *slot* must not be smaller than the radius of the selected milling tool.
- Other feature properties
 - Feature name: *slot*
 - Depended_feature_id: entity_list
 - Domain: manufacturing
 - Nature: negative

Figure 3.26 Machining feature (slot) definition in EXPRESS-G

- Instantiate a *slot* feature

Instantiating a *slot* feature in manufacturing domain often requires automatic or interactive feature recognition with the help of users. Automatic feature recognition is very complicated and is a big research area by itself. In this work, it is skipped. The corresponding features in different domains are associated by the commonly associated final product model. Given the design feature slot shown in figure 3.19, the corresponding machining *slot* feature shown in figure 3.25(a) can be applied. Note that in figure 3.25(a), the part contains two features which are represented as 3D cells, namely, the *slot* and the *base_block* (workpiece). The detailed properties of a manufacturing feature *slot* instance are shown in figure 3.27.

Figure 3.27 The instance of a machining slot feature

3.5 Operation

Owing to the large sizes of CAD files, data transmission among different CAX applications through Internet is very time-consuming and unreliable. It is intolerable for the designer to wait for a long time for updating large-size CAD models across networks. To reduce the network-load, an appropriate way to represent CAD data is needed. Incremental transfer is an alternative way to reduce the network-load. Only the modifications, instead of the whole CAD model, are transferred incrementally during the design process. In this work, an *operation* is used to incrementally transfer model modifications such that the communication load can be reduced.

Operation is defined as a set of related commands which are responsible for functional manipulation on entities. It is directly used to support the interface of a CAX system. As categorized by Chen *et al.* [Chen 2005], operations have two types, namely geometry- and non-geometry-related operations. An operation can be represented in figure 3.28. The geometry-related operations can be further classified into feature-related and low-level operations according to the entities that they manipulate. The low-level operations are to create or to modify low-level entities, such as points, lines and faces. The feature-related operations (feature operation) include instantiating a feature or modifying a feature. Non-geometry related operation could be divided into “auxiliary” and “additional” operations. “Auxiliary” operations are mainly to facilitate the designer in geometric modeling but do not affect the geometric shapes, such as layer management and view manipulation. Other non-geometric operations can be classified into “additional” group, such as those related to file management. Supporting operations is a basic requirement for the generic feature definition; and it is important for the manipulations of the feature model, especially for distributed collaboration over the web because the communication load between distributed client and database server can be reduced to the minimum by communication with operations.

An operation entity has its *name* and *ID*. An attribute named *time_stamp* is used to record time sequence during a collaboration session. An operation records the entities to be created or modified in an *operation_entity_list*. In the *referenced_entity_list*, entities that are related to a particular operation are recorded. For example, when an operation that re-constrains a feature with reference to an element of another feature is sent, the old and new *constrained_entities* and *referenced_entities* are recorded in the *referenced_entity_list* such that the application, which receives this *operation*, can easily match the corresponding entities in its application. Such matched entities in the receiving application are recorded in

the *target_entity_list*, which is used for model update according to the operation. An *operation_rational* specifies what kind of action the operation will do to the operation entity, e.g. for feature-related operation, *operation_rational* specify the actions such as *add*, *delete* or *modify*.

Figure 3.28 Operation representation schema

3.6 Geometrical Representation Schema — Cellular Model

3.6.1 Cellular Model

Unified feature model allows different applications to define features in different ways, but they are associated to the same master product model. As reviewed in chapter 2, explicitly maintaining feature shapes in the product model has many advantages. Therefore, a cellular topology-based geometrical representation schema is adopted in this research. Cellular model represents a part as a connected set of volumetric quasi-disjoint cells [ACIS 2001]. By cellular decomposition of space, cells are never volumetrically overlapped. As each cell lies either entirely inside or outside a shaped cell, a feature shape can be represented explicitly as one cell or a set of related cells in the part. The cellular

model-based geometrical representation schema adopted in this research is shown in figure 3.29.

Figure 3.29 Partial geometry schema with cellular topology [ACIS 2001]

Basically, there are three types of topological entities for cellular topology, which are *CELL*, *CSHELL* and *CFACE*. *CELL* has two subtypes, namely *CELL2D* and *CELL3D*. A *CELL2D* contains a list of *CFACE*s, each of which point to faces that are double-sided and both-outside. A *CELL3D* contains a list of *CSHELL*s. A *CSHELL* represents a connected set of *CFACE*s that bound the 3D region of the cell. A *CELL* is attached to the normal ACIS topology in the *LUMP* level (which represents a bounded, connected region in space, whether the set is 3D, 2D, 1D, or a combination of dimensions). Each *CFACE* has a pointer to a face in the lump and use it in *FORWARD* or *REVERSE* sense.

3.6.2 Boolean Operation

The special characteristic of cellular topology requires for special Boolean operation. This is also directly supported by geometric modeling kernel ACIS.

ACIS provides the capability for performing non-regularized Boolean operation. This is generally implemented by specifying a special argument to a Boolean function. In ACIS, the API function: *outcome api_boolean* has an optional argument *Bool_type* to specify the type of the Boolean operation, which can be *Union*, *Intersection*, *Subtraction*, *Nonreg_union*, *Nonreg_intersection*, or *Nonreg_subtraction*.

Specifying non-regularized Boolean type essentially adds three new conditions to the Boolean operations:

- When *single_sided* faces become *double_sided*, *both_inside* faces, they remain in the resulting body.
- Any face-face coincident region remains in the resulting body.
- No edge or vertex merging is performed at the end of the Boolean.

Due to the first condition, the unite operation always keeps all face regions from the two bodies (although it may split them into separate faces). Due to the second condition, the intersection of two blocks that share a coincident face always leaves the face instead of deleting it. Due to the third condition, subtracting a sheet from a non-coincident sheet leaves the imprint of the subtracted sheet on the other sheet [ACIS 2001].

3.7 Summary

In this chapter, four-layer information infrastructure is proposed to integrate product information in EPM and support multi-view applications by its underlying sub-models. Under unified feature model, a generic feature representation schema is given which includes feature shape representation and constraint representation. The generic feature

model provides a template for different application feature definition. In order to maintain feature relationship, a *depended_feature_id_list* is defined which are used to maintain feature dependency relations (this is to be addressed in chapter 5). In addition, feature labels are also defined to record feature elements and support history-independent model re-evaluation. On the basis of generic feature model and STEP AP 224, design feature and manufacturing feature model are described. Examples (design *slot* feature and manufacturing *slot* feature) are given to illustrate how specific feature type can be defined. In order to reduce communication load between distributed clients and database server during a collaboration session, an operation schema is developed. Finally, detailed geometrical representation schema is investigated based on cellular topology.

CHAPTER 4 DESIGN OF FEATURE-ORIENTED DATABASE

4.1 Introduction

In the previous chapter, four-layer information integration infrastructure and generic feature model has been introduced. To develop a feature-oriented database, the EXPRESS-defined information models must be mapped to database data types for physical storage, workform format for the system development based on ACIS solid modeler via its API functions and an operation-based message in XML format for data communication. The mapping concept is shown in figure 4.1. In this chapter, the first two mapping mechanisms are investigated. For the mapping to the XML format, refer to [Chen 2005]. In this particular project, for the convenience, ACIS SAT format is temporarily used. In this chapter, adopting the proposed mapping mechanisms, the generic feature representation and the geometrical representation schemas in database form are described in detail. On the basis of these database schemas, a feature-oriented database is designed.

Figure 4.1 Information mapping structure

4.2 Mapping Mechanisms

Under the proposed information model structure, the entities at different levels must be mapped into the schema definitions of the workform format (ACIS) as well as the

comprehensive product database such that the proposed unified flexible feature object structure can be represented.

4.2.1 Mapping to Database Schemas

The following are the rules for mapping to database schemas:

- ❑ **Primitive data type.** The mapping of primitive data type between EXPRESS and object-relational database is listed below:

EXPRESS data type	Database data type
<i>Integer</i>	<i>Long or Number</i>
<i>Real</i>	<i>Float or Number</i>
<i>Number</i>	<i>Number</i>
<i>String</i>	<i>Char or Varchar2</i>
<i>Binary</i>	<i>Varchar2</i>
<i>Boolean</i>	<i>Number (0 1) or</i> <i>Varchar2 (true false)</i>
<i>Logical</i>	<i>Varchar2</i> <i>(true false unknown)</i>

Table 4.1 Mapping of primitive data type

- ❑ **ENTITY.** Each entity shall be mapped to an object type. An object table shall be created for each concrete entity.
- ❑ **ENTITY attribute.** An entity attribute shall be mapped to a column with *REF* data type pointing to the referenced object of another object table.
- ❑ **Type inheritance.** By defining object types, inheritance is directly supported in the database. A subtype can be defined under its supertype, and therefore can inherit attributes and methods from the supertype. An example for inheritance mapping is shown in table 4.2.

<div>In EXPRESS: ENTITY feature_label; ID : identifier; Feature_name:string; END_ENTITY; ENTITY face_label SUBTYPE OF (feature_label); Face_name: string; END_ENTITY;</div>	<div>As object type definition Create Type feature_labeltyp AS OBJECT (ID NUMBER(20), feature_name VARCHAR2(30)) NOT FINAL; Create type face_label UNDER feature_labeltyp (face_name VARCHAR2 (30)) NOT FINAL;</div>
---	--

Table 4.2 Example of inheritance mapping

- **Enumeration.** Although enumeration data type is not directly supported, we can simulate it by defining type array or nested tables. Here, an example is shown in table 4.3 where array data type is used.

<div>In EXPRESS: TYPE si_prefix= ENUMERATION OF (exa, pico, deca);</div>	<div>As object type definition CREATE TYPE coordinatetyp as ARRAY(3) OF VARCHAR2(5);</div>
---	---

Table 4.3 Example of enumeration mapping

- **Select.** The *select* data type can be simulated with an attribute to the objects in the select whose real data type is decided by the method of the object type. A method of object type is simple SQL or PL/SQL function (procedure language/structured query language), which is used as a discriminant to decide the select elements' data types. For example, given a STEP declaration:

```
ENTITY cc_design_approval
Assigned_approval : approval;
Items           : SET [1:?] approval_item;
END_ENTITY

TYPE approved_item = SELECT
(product_definition_formation,
```

```
.....
security_classification);
END_TYPE
```

In the database schema, the definition is given in table 4.4. Within this table, the name of the *approval_item* is used to decide from which table its contents shall be retrieved. The definition of *approval_item* such as *product_definition_formation*, *security_classification* need to be defined separately and will not be illustrated here.

Column	Data Type	Description
Part_id	Number	Part number of the enclosing entity
Entity_id	Number	Entity number of the cc_design_approval
approval	number	Approval number
Approval_item	Varchar2	Name of the approval_item

Table 4.4 Example of selects mapping

- ❑ **Aggregate.** Object-relational database supports aggregate data types by the definition of type array and nested tables. Array types are suitable for fixed-size aggregate data type. Nested tables are suitable for aggregations whose sizes are not fixed. For example, the fixed-size aggregate data type *list* is represented as *array* data type in the database as shown in table 4.5.

STEP declaration,	As object type definition
ENTITY APOINT Coordinates : LIST [1:3] OF length_measure; END_ENTITY	CREATE TYPE coordinatetyp as ARRAY(3) OF NUMBER(12,4); CREATE TYPE APOINTtyp as OBJECT(Entity_id OID, coordinates coordinatetyp);

Table 4.5 Example of aggregate data type mapping

4.2.2 Mapping to Workform Format

To support collaborative feature-based modeling, a geometry modeling kernel (ACIS) is incorporated into the proposed architecture to provide lower-level geometrical modeling

services. Therefore, the EXPRESS-defined and extended information models must also be mapped to the data representation schema in ACIS. In such a way the modeler can be tightly integrated with the proposed system to provide API (Application Programming Interface) functions as well as the runtime model.

4.2.2.1 Geometry mapping

The unified feature model allows different applications to define features in different ways, but their definition candidate types, such as geometrical representations, and the corresponding processing methods are common [Tang 2004]. In this research, in order to explicitly maintain feature shape and associative relations in the product model, the cellular model that is directly supported in ACIS cellular husk as shown in figure 3.27, is adopted.

4.2.2.2 Generic feature definition under ACIS framework

ACIS provides an ENTITY-ATTRIBUTE framework, under which we can define user-defined attributes (features, constraints or others). Therefore, it is very flexible for us to extend the ACIS geometric model into a true product model.

In ACIS, part information, which contains geometrical data, features and others, is represented as ENTITIES. An ENTITY is a virtual class; it represents common data and functionality that is mandatory in all classes that represent permanent objects. Any class that represents a permanent model object in ACIS must be derived from ENTITY in order to inherit the common data and functionality that is mandatory in all permanent objects. This is because that each specific class (FACE, LUMP, SHELL, SURFACE, STRAIGHT, etc.), has built the functionality into the related classes to support this determination.

Attributes are a special type of entity used to attach strings, integers or ENTITY pointers to geometric entities; any entity may have zero or more attributes. Attributes can carry simple data, pointers to other entities, or links to application-specific variable length data. Attributes can be attached to any type of entity. Therefore, it is feasible to define features as application-specific attributes in ACIS.

The ATTRIB base class provides the data and functionality that are common to any type of features. The ATTRIB class is derived from the ENTITY class, and performs housekeeping operations to maintain attribute lists attached to model entities. Organization attribute classes are always the first level of derivation from the ATTRIB class. Features for specific applications are derived from the organization attribute classes. Feature derivation structure under ACIS framework is shown in figure 4.2.

Figure 4.2 Feature derivation structure in EXPRESS-G

ACIS attributes can be categorized as simple attributes, complex attributes, bridge attributes and instruction attributes. Following are the rules for defining features, constraints and other attributes in ACIS:

- Use simple attributes to represent properties, e.g. the material of a body, or the color of a face.
- Use complex attributes to represent feature properties such as feature graph nodes, dimensions, tolerance, or constraints.
- Use bridge attributes to link an ENTITY with some application-specific, variable length data.
- Use instruction attributes placed on entities to force certain behavior.
- Aggregate data type shall be defined as ENTITY_LIST. The ENTITY_LIST is an array of ENTITY pointers with an associative variable length. The list provides common functions for the manipulation of its members, e.g. *add ENTITY*, *look up ENTITY* and *[]* operator for accessing list member by position.
- Enumeration data type can be simulated by defining string as the enumeration member or simply using integer data type. For example, domain has the enumeration data type which may be design, manufacturing, assembly and others. It can be defined as follows:

```
static char* domain_name[] = {
    "design",
    "manufacturing",
    "CAE",
    "assembly"
};
```

- Select data type can be simulated in two ways. One is by defining abstract class and defining specific type of the abstract class. The other is to simulate in the same way as enumeration data type.

On the basis of the above mapping rules, a generic feature definition in ACIS format can be expressed as shown in figure 4.3. In the figure, circle point indicates that the two intersecting relationship lines don't intersect at this point.

Figure 4.3 The generic feature definition with ACIS entities

4.3 Schema Definition for Feature-oriented Database

Based on the proposed mapping rules, a feature-oriented database is created in an object-oriented manner in this section.

4.3.1 Geometrical Representation in Database

Figure 4.4 Partial database schemas for geometrical representation

A partial cellular topology-based geometrical database schema is created as shown in figure 4.4. Note that in the schema definition:

- Those attributes with suffix *id* (but without *REF*) represent object identifiers (OIDs), which are the globally unique and immutable object identifier generated by DBMS. An OID allows the corresponding row object to be referred to from other objects.
- Those attributes with *id (REF)* are a kind of built-in data type provided by DBMS which encapsulates a reference to a row object of a specified object type.
- An arrow represents *REF* relationship between object types. For example, in the *COEDGE* table, attribute *edge_id* has the *REF* data type, which is used as a reference pointing to the edge object in the edge table.
- As to abbreviations, 'F' represents first, 'N' represents next, 'P' represents previous.

In the schema definition, not all the surface types and curve types are illustrated in figure 4.4. Instead, plane is used as an example type. With the *surface_type* attribute defined in *SURFACE* object table, different kinds of surface types can be identified. Similarly line is used to represent different kinds of curve types. With the *curve_type* attribute defined in *EDGE* object table, different kinds of curve types can be identified. Such kind of simplification will not affect the validity of this proposed database schema.

In the database schema definition, there exist many lists, which are an aggregate data type. For example, a shell contains a list of faces. In order to consistently manage aggregate data type, the following two methods will be adopted on the basis of mapping mechanisms:

- (a) For ordered list which contains topological-related entities (e.g. a shell contains a list of faces which are topologically related), we follow the way of ACIS native data structure. This is realized by defining link relations in the object data structure,

as illustrated in figure 4.5. In the list owner entity object table, only the first member of the list will be recorded by the entity's ID, which can uniquely identify the first list member in the list member entity object table. Then in the list member entity object table, the next list member in the list will be explicitly recorded by its ID, which is used to identify the next list member. With such a data structure, each member of the list can be identified. Note that the last list member in a list will have a NULL *next_entity_id* pointer.

Figure 4.5 Linked list for ordered list

(b) For unordered list (e.g. a feature contains a list of constraints), the schema shown in figure 4.6 is designed to collect each member of a list from the target object table. Such a list shall be defined as *REF* data type with name *list_id* which refers to list object in the *entity_list* object table by *list_id*. A nested table called *id_list* stores all the list members' ids in the nested table. Within the nested table, *entity_type* is used as a vector to decide from which object table we can get the list members. *Entity_id* uniquely identify entities from entity table. An implicit system generated *nested_table_id*, correlates the parent row object with the row objects in the nested table.

Figure 4.6 Generic schema for non-ordered list

4.3.2 Generic Feature Representation in Database

On the basis of the generic feature representation schema described in figure 3.5, a generic feature representation in database can be expressed as shown in figure 4.7 [Tang 2004].

A feature has *feature_id*, *feature_name*, *part_id* and *domain* as its attributes. The *feature_id* attribute is an OID, which can uniquely identify a feature object in database. Feature_name combined with topological entity name, provide basic indexing for solving persistent naming problem during feature model re-evaluation. *Part_id* specifies which part a particular feature belongs to. *Domain* has enumeration data type, which can be design, manufacturing, CAE and so on. A feature also contains a list of feature elements, a list of constraints and a list of parameters. Here, *feature_element* list refers to feature geometric elements (e.g. cells, faces, edges or vertices that belong to a feature). These feature elements are earmarked by *feature_labels* defined in chapter 3. Each *feature_label* has a *feature_id*, which identifies the owning feature of the label. The *reference_entity_id* can uniquely identify the *referenced_entities* by *Entity_id* stored in entity table (entity here

may be geometrical entity and features). By using feature labels (together with the functional methods of the geometry reference layer in the unified feature model), higher-level feature information can be associated with low-level geometrical entities; and the information model can be established by searching the relevant attributes and re-instate the feature model. Dimensions and tolerances are regarded as a kind of constraint bounded to certain geometrical entities. A constraint of a feature can be uniquely identified by *constraint_id*. *Constrained_entity_list* identifies constrained entity from the entity table by *entity_id* and *entity_type*. The element number of *parameter_list* is not fixed until the specific feature type is determined. For *parameter_list* of a specific feature instance, each list element will be treated as an attribute of the feature.

Figure 4.7 Generic feature representation in database

4.3.3 Product and Part representation in database

Given the geometry and generic feature representation schema, product representation in a feature-oriented database can be created as shown in figure 4.8. As stated in chapter 3, we only focus on the original feature representation of the product and the related parts. Other product-level and part-level related information is not addressed completely here.

Figure 4.8 Product representation in database

A product is identified by *product_id*. A product (the top assembly) may contain a number of subassemblies, parts or components which are assembled with some assembly features. All the assembly features within a module are listed in its *assembly_feature_list*,

which is defined for the top assembly and each sub-assembly of the product. For each assembly feature, a list of *part_ids* (*module_ids*) is used to identify the related member parts (modules) in the part (module) table. An assembly may have new level sub-assemblies which could be referred by *subassembly_feature_id* in the assembly feature table. In the product, subassembly, and part table, *design_feature_list*, *manufacturing_feature_list* and *other_application_feature_list* are stored, which are used to organize feature model for different views at different levels. For example, all design features of a modular subassembly are stored in the associated design feature table. By *feature_id* and *feature_type*, various types of design features can be identified from the specific design feature tables. At the geometric entity level, different kinds of features are built on the basis of the cellular model described in figure 4.4.

4.4 A Simple Case Study

In this section, a simple part is used to illustrate how feature information is represented in the proposed feature-oriented database.

Figure 4.9 A case part

Figure 4.9 illustrates the creation of a simple part with two features, namely a *base_block* feature and a *through_slot* feature. The *base_block* feature is created by two diagonal coordinate points, which are derived from the parameters of the *block*, i.e. length, width, height and its position point. The *through_slot* is positioned by specifying three coplanar constraints. The first constraint is the start face of *slot* that coplanes with front face of *base_block*. The second constraint is the end face of the *through_slot* which is coplanar with the back face of *base_block*. Then, the third is the top face of the feature that is supposed to be aligned with the top plane of the base block. Other parameters such as the length, width and depth of the *slot* are applied as dimensional constraints. Finally, the part is created.

The cellular model of the example part is shown in figure 4.10. As illustrated in the figure, there are two cells (each one contains a cshell) in the cellular model of the part; one for the *base_block* and the other for the *through_slot*. Due to the overlapping of the two cells, three double-sided faces are generated, namely, F_1 , F_2 and F_3 . Each double-sided face has two corresponding cell faces (one contributes to the cell of the *base_block* and the other contributes to the cell of the *slot*). Note that, the cellular model of the example part (got by non-regular Boolean operation) keeps three additional faces (shaded faces with names F_{10} , F_{11} and F_{12}) and two more edges (E_{24} and E_{25}) shown in figure 4.10 in comparison with the traditional B-rep final part geometry obtained by regular Boolean operation as shown in figure 4.11. These extra elements help to maintain the explicit feature shape in the part model. As they have the characteristic *not-on-boundary*, the B-rep evaluation of cellular model can be easily carried out by doing boundary detection and removing entities that are not on the boundary. Mechanism for B-rep evaluation will be described in chapter 5. Note that in figure 4.11, only faces, edges, vertices and two corner points are labeled; other geometrical information (e.g. coedges, loops) is not included.

Figure 4.10 Cellular model of the example part

Figure 4.11 The final B-rep part geometry

Figure 4.12 Database representation of the case part

Based on the proposed database schema, such a part can be represented in database as shown in figure 4.12. In the part table, two design features, namely, *base_block* feature and *slot* feature are recorded with IDs. By *block_ID*, the *block* feature can be recognized from *block* feature table. All attributes of a *block* feature are stored in *block* feature table. Among all the parameters, two positions, which are defined by two vertices (V_{15} and V_{10}), are used to fix the position and orientation of *block*. Each vertex can be identified by *vertex_ID* in the vertex point table. Vertex point table contains all the vertex of the example part. All the feature elements (cell list, face list, edge list and vertex list) of the *block* feature are stored as a list of feature labels identified by *label_IDs*. Due to the space limit, only face elements ($B_{L_0} \sim B_{L_6}$) are taken as an example to explain how feature labels can be used to get low-level feature elements. Other feature elements follow the same way. By the *label_ID* and the corresponding *face_ID* stored in feature label table, the face elements (e.g. $F_0, F_4, F_5, F_6, F_7, F_8, F_9$) can be recognized in the face table. By using the *slot_ID*, the *slot* feature can be identified from the *through_slot* feature table. All the attributes of *through_slot* feature are stored in the feature table. The position field contains a *vertex_ID* (V_4) which uniquely identify a vertex point in the vertex point table. All the face elements of the *slot* feature are stored as a list of feature labels ($S_{L_0} \sim S_{L_6}$). In feature label object table, by the *label_ID* and the corresponding *face_ID*, all the face elements (e.g. $F_0, F_1, F_2, F_3, F_4, F_8$, and F_9) can be recognized from the face table.

In the *slot* feature table, constraints are stored in the *constraint_list*. By the *constraint_type* and *constraint_ID*, different kinds of constraints can be identified from various constraint tables (e.g. *coplanar_constraint* table, *distance_constraint* table). In this case, coplanar constraint with ID C_0, C_1 , and C_2 are stored in the *coplanar_constraint* table. For coplanar constraint with ID C_0 , the constrained feature elements (*constrained_entity*) can be identified by feature and element names. Similarly, the referenced feature elements

(*referenced_entity*) can be identified by feature name, element name. For C_0 , the feature name is *through_slot1*. The element name is F_8 , which point to the face in face table. *Coplanar_constraint* with ID C_1 and C_2 are processed in the same way. Note that in this example, geometrical and topological entities such as shell, face, loop, coedge are stored across different tables, see geometrical representation schema in database in figure 4.4. Note also that figure 4.12 shows only part of the details for geometrical entities defined in the feature. In this way, low-level geometrical data and high-level feature information can be represented in the feature-oriented database.

4.5 Summary

In order to achieve a feature-oriented database to support collaborative product development, mapping mechanisms, from EXPRESS-defined information model to workform format and the targeting database schemas are investigated. Based on the proposed mapping mechanisms, the generic feature representation schema in ACIS is proposed. The generic feature representation schema and the geometrical representation schema in database are also described in detail. Finally, a simple part is taken as the example to illustrate how a feature's information is represented and managed in the feature-oriented database.

CHAPTER 5 FEATURE MODEL EVALUATION AND VALIDATION

5.1 Introduction

After each feature modeling operation (adding, deleting, modifying), the product model must be evaluated. DBMS must be able to manage product feature information effectively and efficiently. This can be achieved by providing mechanisms for model evaluation and validation. History-dependent model evaluation suffers from the problems of high computation cost, large storage space and corruption of feature semantics. Therefore, in this work, a history-independent approach for feature model evaluation is adopted [Bidarra 1999, 2000 and Bronsvort 2004]. The feature precedence relations are dynamically maintained via a feature dependency graph. In addition, the feature geometry validation and constraints solving mechanisms for feature validation are proposed; they have to be supported by geometrical modelers and the corresponding constraint solvers.

5.2 Feature Model Evaluation

5.2.1 Overview

Almost all the existing feature modeling systems are history-based. Model history is stored as a sequence of modeling operations of feature modifications. Feature instances can be added to, modified or deleted from the product model by running the respective feature operations in the model history. Then a new boundary model is created by sequentially re-executing the operations in the modified model history.

However, re-executing the whole model history after modifying or deleting a feature has a high computational cost, and is roughly proportional to the model history size [Bidarra 2000]. One method suggested to improve this is to store all intermediate boundary

representations between history steps. Then, only the history steps after the previous intermediate saving need to be re-executed. However, storing intermediate models between history steps requires a considerable amount of storage space, which is roughly proportional to the square of the model history size. An alternative improvement is to store only the deltas between history steps, and to rollback to the state from which the model needs to be re-evaluated. This requires less storage space, but more computation time. In any of the above-mentioned cases, the steps of re-execution, very often, create or evaluate more features than those actually affected by the modeling operations.

Figure 5.1 Example part

Another shortcoming of history-based re-evaluation of the model is the ambiguous feature semantics due to the chronological feature creation order in the model history. This is illustrated in figure 5.1, where a simple part that consists of a *base_block*, a *blind_hole* and a *boss* in sequence is shown. Note that the *hole* is started from the bottom face of the *base_block*. Later on, the designer modifies the part model by increasing the depth of the *hole* which causes the overlapping between the *blind_hole* and the *boss*. After the modification, the cellular model of the modeling result is shown in figure 5.2. Note that the nature of emphasized overlapping cell shown in the figure is different for history-dependent and history-independent approaches. For history-dependent approach, the overlapping cell is additive, because the last owner of the cell is additive feature *boss* (in

cellular model, the nature of cell is determined by its last owner. For details, please refer to section 5.2.3). Then, an unexpected result after boundary evaluation is obtained; the depth of the *blind hole* is truncated by the *base_block* (see figure 5.3). Obviously, the feature precedence order in the example should be changed from *base_block*→ *blind_hole*→ *boss* to *base_block*→ *boss*→ *blind_hole* in order to get the expected part model shown in figure 5.4. The semantic problem in this case is caused by the static feature precedence order in the model history on which the model re-evaluation algorithm is based. Therefore, in order to maintain the feature semantics, the precedence relations among features should be *dynamically* maintained and updated after each modeling operation. A set of rules are used for this purpose.

Figure 5.2 The cellular model

Figure 5.3 The B-rep model resulted from the history-based evaluation

Figure 5.4 The expected B-rep model

5.2.2 Dynamic Maintenance of Feature Precedence Relations

The global feature precedence relations are determined by dependent feature relations in feature dependency graph and independent feature sequence [Bidarra 2000]. Independent feature here means there is no attachment, positional or other type of constraints between two features. Independent feature sequence here means precedence relation between two independent features.

When a new feature is created, its dependency relations with existing features are explicitly defined in the feature definition; they are recorded in, and can be managed by, the feature dependency graph as explained in section 3.3.2.3 of chapter 3 [Bidarra 2000]. The precedence order of this newly created feature still has to be checked against its independent features by feature overlapping detection. However, the two criteria given by Bidarra can not fully ensure a globally unique precedence relation among all the features of a product model (see examples for rule 4 and rule 5). Therefore, the following rules are proposed hereafter for the dynamic maintenance of the global precedence order of all features.

Figure 5.5 shows how the rules are designed for different cases. By default, the feature precedence relation is determined by the feature creation sequence. Then, feature

dependency relations and independent feature sequence will be checked respectively.

Detailed rules will be discussed in the following part of this section.

Figure 5.5 Feature precedent relation determination

Rule 1 (explicit, [Bidarra 2000])

For two dependent features, if feature f_2 depends on feature f_1 , then f_1 precedes f_2 .

It is easy for us to derive from rule 1 that:

For n dependent features, if:

$$f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow \dots \rightarrow f_n$$

Then, there exist a precedence order which is a sequence of numbers indicating the evaluation priorities of all the features of the model in an ascending order:

$$O_1 < O_2 < O_3 < \dots < O_n$$

Where:

$f_i \rightarrow f_j$: represents feature dependency relation; e.g. $f_1 \rightarrow f_2$ means f_2 depends on f_1 ;

O_i : represents the precedence order number of feature f_i .

$O_i < O_j$: represents the j^{th} feature is ordered after the i^{th} feature.

Rule 2 (explicit)

If a feature depends on two or more features in the feature dependency graph, the precedence order of this feature comes after the depended feature that has the longest length of path (LLP) from the root node in the feature dependency graph. This particular depended feature is called the longest depended feature or LDF.

Here, note that:

A **path** is a graph traversing route from a starting node to a target node, and along the rout, all the nodes traversed are distinct; there could be multiple paths between two nodes.

The length of a path is the number of edges that form the path.

In the feature dependency graph, the LLP of a feature in the feature dependency graph determines the precedence order of the feature. Figure 5.6 shows an example part consists of a *base_block*, a *slot* and two *hole* features. If *hole2* has a distance constraint with reference to *hole1*, the feature dependency graph of the part can be expressed as shown in figure 5.7. Since *hole2* has two depended features (*base block* and *hole1*), the precedence order of the part should be *base_block* → group (*hole1* and *through slot*) → *hole2* according to rule 2. Here, *hole1* is the LDF of *hole2* because *hole1* has the longer length of path compared with the *base_block*. Note that in this example, the precedence relation between *hole1* and *through slot* can be determined by rule 3 (an implicit rule).

Figure 5.6 Multiple depended features

Figure 5.7 The feature dependency graph

The following rules are used to determine feature precedence relations among independent features.

Rule 3 (implicit)

By default, the initial feature creation sequence will be used to determine independent feature precedence relations. In other words, if f_1 precedes f_2 in the feature creation sequence, then f_1 precedes f_2 in their precedence order.

For example in figure 5.6, the precedence relation between two independent features *hole1* and the *through_slot* is not unique (because these two independent features do not overlap and have the same *LLP*. Simply applying rule 4 can not determine a unique

precedence order). Therefore, their precedence relation should be decided by creation sequence. If *hole 1* is created before *through_slot*, then *hole 1* precedes *through_slot*.

The feature creation sequence is defined as an attribute attached to the feature instance. Given a feature that has two or more depended features with the same length of path, the precedence order of these depended features are not unique; in this case, we can group them together and treated them as one element (within the group, the precedence order can be determined by rule 3 as well). Then the feature that depends on the group has the precedence order *just after* the group.

Figure 5.8 Two independent features with no overlapping

Rule 4 (implicit)

For two mutually independent features, if they do not overlap with each other, the precedence relation between them is determined by the LLPs of these two feature; if $LLP_1 < LLP_2$, then $O_1 < O_2$. In the case of $LLP_1 = LLP_2$, the precedence order can be determined by rule 3.

For example, figure 5.8 shows a part that consists of three features (*base_block*, *hole* and *boss*). Both *hole* and *boss* depend on the *base_block* and they are independent. Since the *hole* and the *boss* don't overlap with each other, whether to create the *hole* or the *boss* first will result in the same final part model.

Rule 5 (implicit)

For two mutually independent features with same natures (negative or additive), if they overlap with each other, the precedence relation between them should be determined by LLP of these two features; if $LLP_1 < LLP_2$, then $O_1 < O_2$. In the case of $LLP_1 = LLP_2$, the precedence order can be determined by rule 3.

Figure 5.9 Two independent overlapping features with the same nature

Figure 5.9 (a) and (b) show an example part with a *base_block* and two *boss* features. The modeling operation (moving *boss1*) causes new overlapping of two *boss* features and results in a large *boss*. According to rule 5 and rule 3, the precedence relation between two

bosses (both additive) can be determined by their creation sequence (they have the same LLP, see figure 5.9 (b)). This is the same for the case shown in figures 5.9 (c) and (d).

Rule 6 (explicit)

For two mutually independent features (f_1 and f_2) with different natures, if the overlap of these two features is caused by some modeling operation of f_2 , then feature f_1 precedes feature f_2 [Bidarra 2000].

Taking part shown in figure 5.1~5.4 as an example, the feature creation sequence is *base_block*→*blind_hole*→*boss*. Later on, when the designer want to increase the depth of *blind_hole*, rule 6 will be applied. Although the *blind_hole* and the *boss* have no dependency relation, the modeling operation of increasing *blind_hole* depth causes the overlap of the *blind_hole* and the *boss*. Therefore, the *boss* precedes the *blind_hole* and the feature precedence order is changed to *base_block*→*boss*→*blind_hole*. In this way, the expected modeling result can be obtained as shown in figure 5.4.

```
(For i=1; i<n-1; i++)
{ (for j=i+1, j<n; j++)
  {if (Pxj>Pxi)
    {Xm=Xi;
     Xi=Xj;
     Xj=Xm;
    }
  }
}
```

Here:
X_i represents any feature in the feature set;
X_j represents depended feature of X_i;
Px_i represents the position of feature X_i in the feature map;

Figure 5.10 Algorithm for precedence order generation [Wirth 1976]

Based on the above rules, after each modeling operation, the following algorithm is proposed to dynamically generate globally unique feature precedence relations:

- (1) Find all the features of the part and add them to a list (sorted list for previous feature modeling operation). Feature creation sequence will be used as the default precedence relation between any two features.
- (2) Check dependency relations of this new feature according to the feature dependency graph by applying rule 1 ~and rule 2.
- (3) Check independent feature sequence for this new feature with its independent features with reference to overlapping detection result by applying rule 4~6.

In our application, the feature precedence relations will be generated after each modeling operation. Therefore, the unsorted dataset mentioned in step 1 is in fact sorted dataset with reference to the previous feature modeling operation. Under this situation, the proposed algorithm has the level of complexity $O(n)$ instead of $O(n^2)$.

5.2.3 History-independent Feature Model Re-evaluation

First of all, re-evaluating the feature model requires feature elements (cells, faces, edges and vertices) to be correctly identified in the cellular model. This can be achieved via cellular entity owner list control.

5.2.3.1 Adding a New Feature Instance

This is carried out as follows:

- (1) Create the shape of the new feature (one cell shape) according to feature creation schema, as described in section 3.1.4.2 of this chapter;
- (2) Attach labels of the feature to each face of the feature instance; and

- (3) Carry out non-regular Boolean union operation.

During non-regular Boolean union operation, intersection detection will be carried out for each cell (C_i) in the cellular model against the newly added feature cell (C). Upon cellular decomposition, the owner list of each cell and cell face should be controlled by the following rules [Bidarra 2005]:

- (1) The new cells that are in the intersection of C and C_i are assigned with which are owner list the union of the owner lists of C and C_i ;
- (2) Other non-intersecting cells resulting from the decomposition get their owner lists which are the same as the original cells (either C or C_i);
- (3) The new cell faces lying on the boundary of both C and C_i get the owner list that is the union of the owner lists of the overlapping cell faces from which it originates;
- (4) The new cell faces lying on the boundary of either C or C_i get the owner list from their respective original cell faces;
- (5) The remaining new cell faces get an empty owner list.

Figure 5.11(a) illustrates the creation of a *slot* feature on the *base_block*. The shape of the *slot* is first created as a one-cell shape. Then non-regular Boolean union operation is carried out to create the cellular model of the part. Upon intersection analysis, cell decomposition is performed. On the basis of the above rules for cell and cell face owner list control, the result of modeling operation is shown in figure 5.11(b). Note that there are two cells in the cellular model. One is the original *base_block* cell (has *block* feature in its owner list); the other is a new cell generated by cell decomposition, namely the *slot* cell (which has *block* and *slot* in its owner list). Three double-side faces separate these two cells. Each double-side face has two corresponding cell faces. For example, CF_8 and CF_9

are the two cell faces associated with the top double sides face CF_{10} ; CF_8 is for the *block* cell boundary, while CF_9 is for the *slot* cell boundary.

(a) Base block and slot shape

(b) Result of modeling operation

Figure 5.11 Creation of slot feature on the base block

Note: 'CF_i' represents ith cell face; 'S' means the *slot* feature; 'B' means *block* feature; and '()' refers to the entities in the owner list.

5.2.3.2 Deleting a Feature Instance

This is carried out as follows (assume no other feature depends on the feature to be deleted) [Bidarra 2000]:

1. Traverse through all the cells and cell faces to remove from their owner list the feature to be deleted;
2. Remove all the cells which has empty owner list. This can be realized by removing all one-side faces bounding the cell;
3. Merge the adjacent cells which have the same owner list. This can be realized by removing all double-side faces that separate two cells;
4. Clean up the model by merging the adjacent faces that have the same geometry and whose cell faces have same owner list.

In figure 5.12, to delete the slot feature from the cellular model, all cells and cell faces in the cellular model are traversed through to remove the slot feature from their owner list. The result is shown in figure 5.12(a). Then as two cells have the same owner list, the block feature, these two cells are merged by removing three double-side faces (the underlying faces of CF₈ and CF₉, CF₁₁ and CF₁₂, CF₁₃ and CF₁₄) that separate them. The result is shown in figure 5.12(b). Finally, adjacent faces that have the same geometry and same owner list are merged. This corresponds to the merging of the underlying faces of CF₅ and CF₁₆, CF₇ and CF₁₅, as well as CF₁, CF₂ and CF₁₀. The result is the same as block feature shown in figure 5.11(a) before creating the slot.

5.2.3.4 B-rep Evaluation

The cellular model of a part contains much more information than only the boundary of the final part; it also contains faces that are not on the boundary. Therefore, the B-rep evaluation of the cellular model requires boundary detection of every face in the cellular model. The following rules are adopted to carry out the boundary detection of faces in the cellular model [Bidarra 1998]:

1. For each single-side face of a cell, the nature of cell determines whether it is on-boundary or not. Additive nature of the cell means the face is *on-boundary*; subtractive nature of the cell means the face is *not-on-boundary*;
2. For each double-side face of a cell, if the cell has the different nature with the partner cell that shares the same face, this double-side face is *on-boundary*; otherwise, it is *not-on-boundary*.

Note that the nature of cell is determined by the nature of its last owner in the cell owner list. The sequence of cell owner list is dynamically maintained according to the unique feature precedence order, see section 5.2.2 for detail.

On the basis of boundary detection of each face in the cellular model, the B-rep evaluation of the cellular model can be carried out in the steps as follows:

- (1) Walk through all the faces and find all the faces that are *not-on-boundary*;
- (2) Remove all the cells and the faces that are *not-on-boundary*;
- (3) Merge adjacent faces that have the same geometry.

Also taking the part shown in figure 5.11 as an example, on the basis of rules for boundary detection, the result of boundary detection is shown in figure 5.13. Then, the B-rep evaluation of the cellular model can easily be realized by removing three faces (F_9 , F_{12} and F_{13}) that are *not-on-boundary*. The result is shown in figure 5.14.

Figure 5.13 Boundary detection

Figure 5.14 Result of B-rep evaluation

Note that in figure 5.13 and 5.14, 'O' means *on-boundary*; 'N' means *not-on-boundary*.

5.2.3.5 A Case Study

Figure 5.15 illustrates the creation of an example part which is made up of a *base_block*, a *vertical_support*, a *rib*, a *cylinder* and two *through_hole* features. All the parameters and constraints are listed in table 5.1. Figure 5.16 shows its feature dependency graph.

(a) Creation of *base block*

(b) Creation of *vertical_support*, *cylinder*, *rib* and two *through_holes*

(c) Modeling result

Figure 5.15 A case part

feature	Constraints and parameters
<i>Baseblock</i>	<p><i>Determined by two position points (0,0,0) and (100,100,10).</i></p> <ol style="list-style-type: none"> 1. <i>length of baseblock = 100;</i> 2. <i>width of baseblock = 100;</i> 3. <i>height of baseblock = 10;</i>
<i>Vertical_support</i>	<ol style="list-style-type: none"> 1. <i>Geometric constraint: verticalsupport_start coplane with baseblock_left;</i> 2. <i>Radius of arc C_2 $R_1=16$;</i> 3. <i>Center of arc C_2 determined by two distance constraints:</i> $D_1=50$; $D_2=60$; 4. <i>C_1 tangent to C_2 ;</i> 5. <i>C_3 tangent to C_2;</i> 6. <i>Extrusion length $W_2=10$</i>
<i>Cylinder</i>	<ol style="list-style-type: none"> 1. <i>Geometric constraint: cylinder_top coplane with baseblock_left;</i> 2. <i>Center of cylinder_top concentric to arc C_2 ;</i> 3. <i>Height of cylinder $H = 85$;</i>
<i>Rib</i>	<ol style="list-style-type: none"> 1. <i>distance constraint: distance between C_5 and C_7:</i> $D_3 = D_2 - (R_1 - \sqrt{R_1^2 - (W_1/2)^2})$ 2. <i>Extrusion length $W_1=10$;</i>
<i>Through_hole1</i>	<ol style="list-style-type: none"> 1. <i>Geometric constraint: through_hole1_top coplane with cylinder_top;</i> 2. <i>Geometric constraint: through_hole1_bottom coplane with cylinder_bottom;</i> 3. <i>Geometric constraint: Center of through_hole1 concentric to the center of cylinder;</i> 4. <i>Radius of through_hole1 = 8;</i>
<i>Through_hole2</i>	<ol style="list-style-type: none"> 1. <i>Radius of through_hole2 $R_3 = 3$;</i> 2. <i>Through_hole2_topcenter determined by three distance constraints D_1, D_2+R_1, and D_4 ;</i>

	3. Height of through_hole2 $H_2 = R_1 - \sqrt{R_2^2 - R_3^2}$
--	---

Table 5.1 Features, constraints and parameters in the example part

The precedence order of the part can be generated according to the rules described in section 5.2.2 as follows:

Base_block → vertical_support → cylinder → rib → through_hole 1 → through_hole 2

Figure 5.16 Dependency graph of the initial case part

In the later design stage, for manufacturing reason (the drilling of the hole is better from a planar surface), the designer needs to add a *cylinder_boss* feature. He can simply add the *cylinder_boss* on the top of the *cylinder* overlapping with the *hole2* as shown in figure 5.17. Then, feature overlapping detection and semantic constraint checking will be carried out. The semantic constraints here refer to that *through_hole2* must have both top and bottom faces *not_on_boundary*. In this case, a constraint conflict happens because the semantic constraint of the *through_hole2* can not be satisfied if the current precedence relation is kept. Therefore, a hint will be generated by the system to inform the user how to solve such problem. There could be two options. Option I is to change the precedence order by redefining these two features; Option II is change only the *through_hole2* from *through_hole* to *blind_hole*). In the first case, two features are redefined as follows:

feature	Constraints and parameters
Cylinder boss	<div><div>1. Radius of cylinder boss $R_4 = 6$;</div><div>2. The top center of cylinder boss determined by three distance constraints: D_1, D_4 and D_5 (distance between top center of cylinder boss and top of base block in Z axis;</div><div>3. Height of cylinder boss $H_2 = D_5 - D_2 - \sqrt{R_1^2 - R_4^2}$</div></div>
Through_hole2	<div><div>1. Radius of through_hole2 $R_3 = 3$;</div><div>2. Top center of through_hole2 coplane with top center of cylinder boss ;</div><div>3. Top center of through_hole2 concentric with top center of cylinder boss;</div><div>3. Height of through_hole2 $H_2 = D_5 - \sqrt{R_2^2 - R_3^2}$</div></div>

Table 5.2 Redefine two features

Figure 5.17 After adding a boss for case I

After modification, the feature precedence order of the part will be changed from
Base_block → *vertical_support* → *cylinder* → *rib* → *through_hole1* → *through_hole2* →

boss to *Base_block* → *vertical_support* → *cylinder* → *rib* → *through_hole1* → *boss* → *through_hole2*. Therefore, the result part model after modification can be generated as shown in figure 5.17. The dependency graph of the modified part can be expressed as shown in figure 5.18.

Figure 5.18 Dependency graph for case I

For the second case, *through_hole2* is changed from through to blind. The corresponding modeling result and feature dependency graph is shown in figure 5.19 and figure 5.20. Note that the second option is reasonable for design model. But it is not applicable for manufacturing purpose because the access plane for drilling the *hole* is blocked.

Figure 5.19 After adding a boss for case II

Figure 5.20 Feature dependency graph for case II

Figure 5.21 After removing the boss feature

Later on, the designer wants to remove the *boss* feature. According to the feature dependency graph shown in figure 5.18 and the latest feature precedence order, the removal of *boss* feature can be done by removing the boss as well as the *hole* feature that depends on the *boss* feature. Then the part after removing the *boss* is shown in figure 5.21 and its dependency graph is shown in figure 5.22. After this step, by default, the *hole* feature that has just been deleted can be re-instate and the result will be the same as shown in figure 5.15(c). Actually, it is also possible to give user a choice to delete the *hole* feature permanently, then the result shown in figure 5.21 will be the final.

Figure 5.22 Feature dependency graph after removing boss feature

5.3 Maintenance of Feature Validity

Feature validity must be checked during feature modeling operations in order to maintain the feature semantics. A feature is valid so long as the feature satisfies all the relevant constraints (whether local or inter-application ones) and the feature geometry is valid. Mechanisms for feature validation are explained in figure 5.22 which include feature geometry validation and constraint solving.

5.3.1 Validity of Feature Geometry

With the incorporation of a solid modeler, feature geometry validation can be managed by the feature geometry manager. It calls modeler-provided functions. For example, assume function *is_BODY()* is defined in such a way that it takes any geometric entity as an argument and returns *TRUE* if the entity is a BODY, *FALSE* if it is not. In this way, the details of geometrical operations are readily looked after by the solid modeler; hence, the development effort is significantly reduced.

Figure 5.23 Maintenance of feature validity

5.3.2 Constraint Solving

In this work, Dohmen’s [Dohmen 1998] approach is adopted for constraint solving. As shown in figure 5.23, all kinds of constraints are managed by constraint manager in a constraint graph for EPM, which contains sub-graphs for different applications (e.g. design application, manufacturing application). The constraint graph contains a symbolic representation of the feature models of the different views in terms of constraints.

Therefore, constraint-solving procedure after each feature modeling operation includes the following two activities:

- 1. Local constraint solving for the editing application;
- 2. Constraint propagation to other applications.

5.3.2.1 Local Constraint Solving

Figure 5.24 Constraint solving procedure [Dohmen 1998]

Dohmen [Dohmen 1998] proposed to solve different kinds of constraints with different kinds of constraint solver, e.g. using SkyBlue [Sannella 1993a, 1993b] solver to solve algebraic constraints, using degrees of freedom analysis [Kramer 1992] to solve geometric constraints. After each modeling operation, local constraint solving will be carried out in steps as shown in figure 5.24.

Firstly, algebraic constraint graph (sub-graph of the main constraint graph in the edit application) will be updated by adding new algebraic constraints and passing modified feature parameters.

Secondly, algebraic constraint solver (SkyBlue) will be called to solve the algebraic constraint graph, after which the main constraint graph is updated according to the newly determined parameters.

Then, geometric constraint graph (sub-graph of the main constraint graph) is updated on the basis of newly updated main constrain graph, followed by calling geometric constraint solver (degrees of freedom analysis) to solve geometric constraints. After that, the solved geometric constraint graph is mapped back to the main constraint graph such that the parameters, geometric entities, positions and orientations of newly added feature or modified feature can be determined.

Next, the feature parameter values are checked against the dimension constraints. Then, the product geometry is updated.

Finally, the semantic constraints are checked.

Algebraic constraints are used to specify explicit feature parameter values; for implicit parameters, a value determined from an algebraic constraint would always be evaluated by the geometric solver. However, implicit shape parameters can be used as input values in algebraic constraints, by which other parameters are determined.

Figure 5.25 Constraint graph of the example part

For example, in an assembly view, the length of a pin in one part could be declared to be equal to the depth of a through-hole in another part. The constraint manager handles this situation in two iterations. In the first iteration, only the algebraic constraints with explicit parameters are solved. Then, after solving geometric constraints, the implicit parameters are determined and constraint graph is updated. In the second iteration, the algebraic constraints on these implicit parameters are included and solved. This is depicted as the backward arrow in figure 5.24. This can cause a value change of other, explicit parameters, and therefore the geometric constraints are re-solved.

Figure 5.26 Updated constraint graph for the example part

Use the same example as shown in figure 5.21. The constraint graph of the example part in design view can be expressed as shown in figure 5.25. Then, in the later design stage, a *cylinder boss* and a *through_hole* are added. For adding a *cylindrical boss*, a *cylinder* feature will be instantiated by the designer. By *offset* constraint (see constraint graph

shown in figure 5.26), the Z coordinate of top of cylinder can be determined. Then, two *distance* constraints (one with reference to the front of the *base_block*, the other with reference to the bottom of *cylinder*) decide the X and Y coordinate of the top center of the *cylinder_boss*. Finally, after giving a fixed value of *cylinder* radius, an algebraic constraint determines the height of the boss (by deriving height of the boss from radius of *cylinder* and *cylinder_boss*, see table 5.2) after which the shape and position of *cylinder_boss* feature is determined.

The adding of a *hole* feature is carried out by: first, solving algebraic constraint by determining the radius of *hole* according to the radius of *cylinder_boss*. Then, after updating the constraint graph, the position and orientation of *hole* are decided by solving geometric constraints (*coplanar* constraint to decide top of the *hole*, *concentric* constraint to fix the top center of the *hole* and *algebraic* constraint shown in table 5.2 to determine bottom of the *hole*). Finally, the *hole* feature is created given the shape and position of the hole feature. The final part is shown in figure 5.17 and the updated constraint graph is shown in figure 5.26.

This constraint solving approach solves constraints in two steps. Firstly, algebraic constraints are solved by SkyBlue constraint solver. Then, the geometric constraints are satisfied. If after that there are unsatisfied algebraic constraints left, the two steps are repeated. In this way, the dedicated solvers can work independently. Such an approach has the advantage of modularity, making it possible to deploy other constraint solvers.

5.3.2.2 Multi-view Constraint Propagation

After solving constraints in one view, the change should be propagated to other related views. Such kind of propagation can be realized by inter-application constraint (or link

constraint [Dohmen 1998]) evaluation. Inter-application constraints link views of a product via detailed geometry entities.

Inter-application constraints can be derived from generic constraint model described in chapter 3. Inheriting all properties from the generic constraint, an inter-application constraint has *linked_views* as its proprietary property to specify different views linked by it. Inter-application constraints may have various types, e.g. *coplanar*, *collinear*, and *coincident*, etc. In this proposed feature-oriented database, all inter-application constraints are managed in the inter-application constraint table, see table 5.3. The detailed constraint contents are managed in the constraint table of specific constraint type and are identified with *constraint_id*.

During a local constraint solving process, inter-application constraints are deactivated. Only when the edit view is updated in the database, are these inter-application constraints checked because database triggers the process when the edit view is updated.

Column	Data Type	Description
Constraint_id	OID	ID of the inter-application constraint
Owner_list_id	REF	ID reference to the owner.
Linked_view_list	Varchar2	Specify the views linked by the constraint
Constraint_type	Varchar2	Type of constraint

Table 5.3 Inter-application constraint table

Database triggers are designed at row objects level. Upon checking in one view feature model, the following circumstances will occur:

- (1) If no inter-application constraint exist, return;
- (2) If inter-application constraints exist but no modification of the current model, return;

- (3) If inter-application constraints exist but the *unique referenced_entity_id* has not been changed, the linked view will try to propagate changes according to related inter-application constraints and re-solve constraints in the linked view;
- (4) If inter-application constraints exist and the *unique referenced_entity_id* has been changed, database will inform the user of this linked view to regenerate model according to the updated feature model.

The following example shown in figure 5.27 illustrates how inter-application constraints are used to link different views.

As shown in figure 5.27, the example part can be represented as a *base_block* with two *extrusions* and two *blind_holes* in design view. In manufacturing view, the part can be manufactured by: (1) choosing a *stock*; (2) then endmilling the top *planar surface*; (3) endmilling the *slot* feature; and (4) drilling two *blind_hole* features. As the *slot* feature in manufacturing view share the same faces with design view (e.g. *through_slot_left* with *extrusion_1_right*, *through_slot_right* with *extrusion_2_left* and *through_slot_bottom* with *base_block_top*), five inter-application constraints, which are $I_C_1 \sim I_C_5$ as shown in figure 5.27, can be established. I_C_1 links two feature elements, namely *extrusion_1_right* in design view and *through_slot_left* in manufacturing view, by shared face F_1 in the B-rep model. Similarly, I_C_2 links *extrusion_2_left* in design view with *through_slot_right* in manufacturing view by F_2 ; I_C_3 links *base_block_top* in design view with *slot_bottom* in manufacturing view by F_3 ; I_C_4 links *extrusion_1_top* in design view with *base_block_top* in manufacturing view by F_4 ; and I_C_5 links *extrusion_2_top* in design view with *base_block_top* in manufacturing view by F_5 . By defining inter-application constraints, the change in one view can be automatically propagated to the other views. In the example shown in figure 5.27, if the width of *extrusion_1* is changed, by these inter-application

constraints and two local constraints (coplanar constraint to define the start and end of *slot*) in manufacturing view, the width of the slot can be automatically changed. Similarly, the modifications in the manufacturing view can be propagated back to the design view by inter-application constraints. It can be seen here that inter-application constraints have to be bi-directional links.

Figure 5.27 Different views linked by inter-application constraints

Note in figure 5.27, to make the figure clear, only the removal volume of the manufacturing feature is illustrated as 3D cells. The accessibility volume of the manufacturing feature is not illustrated. In the figure, 'I_C' represents an inter-application constraint, and 'F' represents a face.

5.3.2.3 Constraint Conflict management

In this research work, only well-constrained and over-constrained situations are considered. To solve constraints conflicts (over-constraint situation), two constraint priorities are specified in the generic feature definition (see chapter 3).

- *Constraint_strength*: Constraints of a specific application may have different priorities. It has an enumeration data type, which may include several levels, such as required, *strong*, *medium* or *weak*. It represents the extent that the constraint needs to be imposed when constraints conflict with each other. If conflict of local constraints occurs, local constraints solver can determine automatically which constraint should be satisfy first according to the value of *constraint_strength*.
- *Domain*: Its value can be predefined which regulates priority sequence of different domains, or is set by an authorized user. Any conflict of inter-application constraints will be detected by constraint manager after which the constraints solver can trigger corresponding applications to reevaluate the product model according to *domain*.

Only when all constraints are checked and the feature geometry is validated, does feature validation finish.

5.4 Persistent Naming

As stated in chapter 3, persistent naming problem originated from the dual representation of solid objects for most current CAD systems: on the one hand parametric feature definition and on the other hand B-rep model. During feature modeling operation, intermediate B-rep entities can be used in the subsequent parametric definition. These B-

rep entities are not persistent because they can be split, merged or deleted in the modeling operation. As a result, parametric definition referring to these entities may be ambiguous or invalid [Bidarra 2002]. On the other hand, the new solid modeling systems cannot guarantee that the parametric models are valid and unambiguous, and the results of modeling operations are not always predictable [Raghothama 1998].

A lot of effort has been made to solve the problem, cellular topology-based persistent naming [Bidarra 2002] and a semantic ID naming schema based on geometry continuity [Wang 2005]. All these works helped to obtain a better understanding of the issue. In this research, under the unified feature model, a geometry_reference layer, which is used to connect higher-level feature information with low-level geometric entities, is created on the basis of generic naming schema proposed by Capoyleas *et al.* [Capoyleas 1994, 1996 and Chen 1996]. The reference layer assigns a unique name to each topological entity by using a combination naming method by using topology-based, local- and feature-orientations. Then this reference layer uses a mapping mechanism to match an entity in the new model with one or more entities in the old model. Whenever designer wants to modify the high-level feature specifications, he submits a reference name to the reference layer; the reference layer will return one or more corresponding topological entities to him as long as this entity has not been deleted completely.

5.5 Summary

In this chapter, rules for the precedence order determination of dependent and independent features are proposed on the basis of Bidarra's work. An algorithm for dynamically generating feature precedence relation is given. On the basis of dynamically generated feature precedence, the mechanisms for history-independent feature model re-evaluation are proposed, analyzed, illustrated by a case study. Furthermore, a graph-based

approach [Dohmen 1998] for constraint solving is adopted to maintain feature validity in the feature-oriented database.

CHAPTER 6 DESIGN OF A COLLABORATIVE SYSTEM

6.1 Introduction

As discussed in the previous chapters, information integration is the most important aspect for collaborative product development and a feature-oriented database is ideal to support flexible information sharing. With the generic feature representation schema, the database schema, and the mechanisms for feature model re-evaluation and validation, it can be appreciated that the product lifecycle information across different engineering applications can be well managed by using a feature-oriented database. However, the conventional under-one-roof product development is no longer able to deal with a competitive and fast changing world market; collaborative product development in a global environment has become the industry trend. It has been envisaged by market analysts that a network-based multi-view engineering collaboration system is demanded by the industry. Unified feature-based modeling approach and the corresponding methodology (future work) can be very useful when a collaborative product development system is to be developed. In this chapter, a database-driven, computer network-based collaborative feature modeling system is designed and prototyped to enable information sharing among CAx applications. The major functional modules in the proposed system are described in detail in the following sections. Then, with a typical case, the prototype system is illustrated. The algorithms for database management as well as the control mechanisms for data concurrency and security are explained.

6.2 System Design

6.2.1 System Requirements

The requirements of distributed collaborative engineering systems are identified as follows:

- ❑ Such a system should be compatible to the existing STEP standard as much as possible, and it should be aimed to evolve into a extended international standard in the long run because such a standard would allow the participation by a wide variety of parties, including suppliers and customers.
- ❑ The system has to be scalable from small to large enterprises in real applications. A DBMS should be used to manage the storage and retrieval of the large amount of information. The DBMS should also allow for the ready integration with other functions within or across the established or virtual enterprises.
- ❑ It has to be able to support the current interactive 3-D graphical functions for the representations of parts and products. Typical engineering context should be available for users to manipulate different aspects of the product lifecycles.
- ❑ The data representation and process data structures should be compatible to the main stream Internet and visualization standards.
- ❑ Feature information managed in engineering database should have levels of granularity (not in file format).
- ❑ Feature information should be shared among multiple applications such that multi-application links can be established.

6.2.2 System Framework

According to the system requirements, the web-based, database-driven and feature-oriented system architecture [Tang 2004a] has been proposed to enable information sharing among CAX applications as shown in Figure 6.1. The proposed system adopts client-server architecture which includes client, application servers and database server. The application servers include a web server, an application object server and a feature object server to provide different functionality. In the following section, the roles of these functional modules are described in detail.

6.2.2.1 Web Server

The web server contains a collaboration manager, security manager and session manager. Collaboration manager provides two kinds of accesses for multiple users, namely, “HTTP” access through MDAI (multi-view data access interface) by adopting ASP (active server pages) and direct socket-based access. Therefore, it can support data check-in and check-out by multiple users through the web; it also supports socket-based collaboration among multiple users. The security manager is used to check whether the user has the right to access the product model data and what kind of access right he may have. Users are classified into several groups. Each group has different access right. Such management data are stored in the database. When a user signs in through collaboration manager by his user name and password, the security manager will check the user’s information stored in the database. Then according to the role of the user (designer role, manufacturing role, CAE analyzer role and so on), selective views of the product model data will be prepared for the user. The session manager is responsible for controlling concurrent access by multiple users of the same data, and controls concurrent access of product data according to *domain_strength* attributes of the users. The details of security control will be discussed in section 6.4.

6.2.2.2 Application Object Server

The application object server provides different application packages (e.g. design or manufacturing tools) for different users such that users can interactively carry out feature-based modeling operations with the support of the feature object server. The product model manager is responsible for managing different application views as an integrated product information model, and maintaining the information consistency in the product level. The product information model includes different feature models with constraints, inter-application constraints and the underlying solid model (cellular model).

Figure 6.1 Overall system architecture

6.2.2.3 Feature Object Server

The feature object server maintains different application feature libraries and therefore can provide feature objects (refer to chapter 3 for more details) for application packages. To maintain the meaning of a feature during feature modeling operations, such as adding,

deleting and modifying features, the feature manager will call the constraint solver and the geometrical modeler to create/update all feature instances based on those feature creation/manipulation functions and constraint solving functions defined in application-specific feature class. The constraint solver can check the validation of all constraints, which are part of the feature definition. The geometrical modeler here is used to validate feature geometry. The mechanisms for feature model evaluation and validation have been described in detail in chapter 5.

6.2.2.4 Database Server

The database server provides physical storage for all kinds of data including product model data, security management data and so on. Within the database, geometrical data and features for different applications are stored as data elements across tables. Detailed geometrical representation schema and generic feature representation schema can be generated based on the mapping mechanisms explained in chapter 3 under unified feature modeling framework. Database manager can provide data manipulation functions (e.g. save, restore and validate functions) with the help of geometrical modeler. These functions are used to organize information for different application views according to users' requirements.

6.2.2.5 Geometrical Modeler

In the proposed system, the solid modeler has been tightly integrated in four layers in order to manage product and process information [Tang 2005], see figure 6.2.

First, its API functions are called constantly which are encapsulated within the feature manipulation methods during the collaboration sessions between the end users and the application server. These functions include geometry construction functions, modification functions, and computation functions for creating, modifying and interrogating solid models. Therefore, higher-level feature modeling functions (e.g. *create_feature()*,

modify_feature()) can directly call these modeler-provided functions. For example, a *create_feature()* function calls geometry construction functions (e.g. *api_solid_block()*) provided by geometrical modeler to construct feature geometry. A feature validate function (e.g. *feature.is_feature()*) call modeler-provided functions (e.g. *face.is_face()*) to check if the feature geometry is valid.

Figure 6.2 Integration of modeler with feature-oriented database

Second, all the geometrical entities are manipulated and their run-time consistency maintained through the solid modeler’s implicit runtime data structure module. By the proposed mapping mechanisms, from EXPRESS-defined information model to modeler workform format and the target database schemas, data consistency in the feature-oriented database can be maintained using modeler functions.

Third, it also provides runtime functional support directly to the end users via commands dynamically.

Fourth, the solid modeler has also to support the repository operations via the DB manager. This will be further explained in section 6.3.

6.2.3 Information Flow

In the proposed system, when a user wants to check-in or check-out product information, he can access through ASP. Then, the security manager, by connecting to the database manager, will check the user information stored in database, and decide whether the user has an access right, what kind of access right he has, and what kind of information (application-specific view) he wants. Finally, he can check-in or check-out product information. If multiple users want to collaboratively design a product, the participants have to register first with the collaboration manager. After security checking, a new session is established according to the user's profile. Similarly, multiple users can work collaboratively work on the product through the collaboration server. During modeling collaboration, feature level operations are used in most cases to minimize communication load among distributed clients and server. The collaboration manager, through product model manager, consistently manages product information in the server side. After each feature operation, such as insertion, modification and deletion, the geometrical modeler will be called to validate the feature geometry. Then the feature manager will call constraints solver to check all the feature constraints (intra-application constraints and inter-application constraints) to validate the feature model. Finally, the finished product model can be stored into the database by the database manager.

6.3 Prototype the Proposed System

6.3.1 The Case

In this section, the design of a vise product is taken as an example to illustrate the whole prototyped system. The vise, which is a kind of fixture, is used to fix manufactured part. It contains multiple components as shown in figure 6.3. The turning of the handle drives the

moving_jaw along the axis to or away from the *fix_jaw* such that two jaws can fix or loosen the work piece.

Figure 6.3 Assembly model for the vise product

6.3.2 Feature-oriented Database

In the prototyped system, the feature-oriented product database has been established on the basis of database schemas (generic feature schema and geometric representation schema) described in chapter 4. For database server, ORACLE 9i, an object-relational database is adopted. The established database schema is shown in ORACLE database manager, see figure 6.4. As addressed in chapter 4, product information, which includes geometrical data, features (with parameters and constraints) and other information, is stored across multiple database tables. High-level feature information connects with low-level geometrical data with the help of intermediate reference layer (feature labels). The use of database makes information sharing among multiple applications with great

flexibility and granularity. Currently, some design features and manufacturing features are defined under the generic feature model and therefore can be managed in the feature-oriented database. For other application view features, more research work and implementation work are needed.

Figure 6.4 The database tables created in the prototype

Figure 6.5 (from ORACLE SQL* session) shows a portion of the geometrical data (faces) stored in the database, which belongs to the part *fix_jaw*. Figure 6.6 lists all the design features of part *fix_jaw*. Note that for design feature model of *fix_jaw*, the relations among features, parameters and constraints are shown in figure 6.7, which is the constraint graph. As described in chapter 4, all these relations are implemented in the feature-oriented database by using OIDs.

PART_ID	SOLID_ID	NEXT_FACE_ID	PLANE_ID	LOOP_ID	FACE_ID	SENSE
fix_jaw	0	8	271	35	7	0
fix_jaw	0	9	272	37	8	0
fix_jaw	0	10	273	39	9	0
fix_jaw	0	11	254	41	10	0
fix_jaw	0	12	255	43	11	0
fix_jaw	0	13	274	44	12	0
fix_jaw	0	14	256	45	13	0
fix_jaw	0	15	257	46	14	1
fix_jaw	0	16	257	48	15	1
fix_jaw	0	17	258	50	16	1
fix_jaw	0	18	259	51	17	0
fix_jaw	0	19	260	52	18	0
fix_jaw	0	20	261	53	19	0
fix_jaw	0	21	262	54	20	1
fix_jaw	0	22	263	55	21	1
fix_jaw	0	23	263	57	22	1
fix_jaw	0	24	264	59	23	1
fix_jaw	0	25	265	60	24	1
fix_jaw	0	26	266	65	25	1
fix_jaw	0	0	267	66	26	0
fix_jaw	0	5	269	29	4	0
fix_jaw	0	6	253	31	5	0
fix_jaw	0	4	268	27	3	0

24 rows selected.

Figure 6.5 Partial geometric data stored in database

PART_ID	SOLID_ID	FEATURE_ID	FEATURE_NAME	OWNER_ID	FEATUERTYPE
fix_jaw	0	356	wedgcut0	0	9
fix_jaw	0	357	slot0	0	0
fix_jaw	0	358	counter_bore_hole0	0	10
fix_jaw	0	355	extrusion0	0	8
fix_jaw	0	359	counter_bore_hole1	0	10
fix_jaw	0	360	simple_hole0	0	2
fix_jaw	0	361	simple_hole1	0	2

7 rows selected.

Figure 6.6 The feature list of *fix_jaw* part

Figure 6.7 Constraint graph of *fix_jaw* in design view

6.3.3 Database Management

As shown in figure 6.1, in the prototyped system, database manager (DB manager) is responsible for reorganizing data elements stored in different database tables for different applications, such that the proposed system is multi-view supported. In order to manage product and process information, which includes feature information, geometrical data and

other information, generic *save()* and *restore()* functions have been developed to organize information for different application views according to users' requirements.

In order to save and restore product information, DB manager must get connected with database server via OCCI (Oracle C++ call interface), which serves as the bridge (see figure 6.2). OCCI is an API that provides C++ applications access to data in an Oracle database. OCCI enables C++ programmers to utilize the full range of Oracle database operations, including SQL statement processing and object manipulation.

In comparison with other common database drivers, such as ODBC (Open database connectivity) and JDBC (Java database connectivity), OCCI takes fully advantages of ORACLE object features. OCCI enables:

- ❑ High performance applications through the efficient use of system memory and network connectivity;
- ❑ Scalable applications that can service an increasing number of users and requests;
- ❑ Comprehensive support for application development by using Oracle database objects, including client-side access to Oracle database objects;
- ❑ Simplified user authentication and password management;
- ❑ n-tiered authentication;
- ❑ Consistent interfaces for dynamic connection management and transaction management in two-tier client/server environments or multi-tiered environments;
- ❑ Encapsulated and opaque interfaces.

Figure 6.8 *Save()* algorithm

6.3.3.1 *Save Algorithm*

During the saving process, the solid modeler has to extract all the information (geometrical entities, topological entities and features with constraints) from its runtime data structure and then save them into the database, see figure 6.8. Save algorithm can be expressed in the steps as follows [Tang 2004b]:

- (1) Initiate algorithm by selecting the part and creating an empty entity list;
- (2) Add the part to be saved to the empty entity_list and extract all the entities (features, geometrical entities such as line, plane as well as topological entities such as solid, shell, face, and so on) to be saved from the part. ACIS API functions, e.g. *api_get_lump()*, *api_get_face()* are used;

- (3) Add all these entity pointers to a graph map so that object pointers can be fixed as unique database Object Identifiers (OIDs). The map type for `save()` procedure, which is used to convert ACIS object pointers to unique database OID, is defined as follows:

```
typedef ENTITY* key;
typedef ENTITY* value1;
typedef int value2;
typedef map<key ,pair<value1,value2> > dat_typ;
dat_typ dat;
key k;
value1 v1;
value2 v2;
dat_typ::const_iterator pos;
```

- (4) Use such object pointers to call `save()` functions of the specific class (e.g. `point.save ()`, `vertex.save ()` or `feature.save()`) to save part data to the database.

With this proposed `save()` algorithm, geometrical data as well as feature information can be saved to database according to user's requirement.

6.3.3.2 Restore Algorithm

Restore algorithm, illustrated in figure 6.9 has the following steps [Tang 2004b]:

- (1) All the entities of a part are retrieved from the database by searching their linked object IDs;
- (2) Reconstruct new objects, e.g. features; geometrical entities such as lines; as well as topological entities, such as faces. Upon reconstruction, all the objects will be validated;

Figure 6.9 *Restore()* algorithm

- (3) Add all the entities to a newly generated object graph map and convert these object IDs to genuine pointers. The map for restore procedure, which is responsible for converting unique database OID to genuine object pointers, is defined as follows:

```

typedef int key1;

typedef int index1;

typedef ENTITY* index2;

typedef map<key1 ,pair<index1,index2> > dat1_typ;

dat1_typ dat1;

key1 k1;

index1 ind1;

index2 ind2;

dat1_typ::const_iterator pos1;
  
```

- (4) Create an entity list and add all the entities to the list to form the part. Validation, e.g. geometry and feature validation will be carried out during this procedure.

On the basis of the proposed *restore()* algorithm, geometrical data and feature information can be retrieved from database according to user's requirement.

6.3.4 Data Concurrency and Security Control

6.3.4.1 Concurrency Control

As the proposed system is supposed to support shared accesses for multiple users, data integrity of EPM must be maintained while enabling maximum concurrent access to the data. Concurrent access of data at the lower level is controlled by the database locking mechanism and managed at higher-level by a session manager.

Database locks are used to control concurrent accesses to a data object and prevent from destructive interactions between users accessing data. Oracle supports two types of locks as stated below:

❑ Automatic Locking

By default, Oracle lock is performed automatically and requires no user action. Implicit locking occurs for SQL statements as necessary, depending on the action requested. Oracle's lock manager automatically locks table data at the row level. By locking table data at the row level, contention for the same data is minimized. Oracle's lock manager maintains several types of row locks, depending on what type of operation established the lock. In this automatic locking mechanism, there are two types of locks: exclusive locks and share locks. Only one exclusive lock can be placed on a resource (such as a row or a table); however, many share locks can be placed on a single resource. Both exclusive and share locks always allow queries on the locked resource but prohibit other activity on the resource (such as updates and deletes). Oracle's implicit locking mechanisms lock data at

the lowest level of restrictiveness to guarantee data integrity while allowing the highest degree of data concurrency.

□ Manual Locking

Default locking can also be override under some circumstances. Oracle allows manual locking to override automatic locking features at both the row level (by first querying for the rows that will be updated in a subsequent statement) and the table level.

A session manager control the database locks through the database lock manager according to predefined *domain_strength*. There are two kinds of sessions, *view only session* and *edit session*. If a user executes a view only session, he can execute as many queries as he like against any tables. Other users of the data do not need to wait for him. When a user creates an edit session to do modeling operation, all related tables are locked at the row level. Therefore, the session of any other users who want to use the same data will be suspended by the session manager before the ongoing session finishes, unless the user has higher *domain_strength* (if so, he can send request to session manager to apply for the control of the data). Following are the rules for concurrency control in the prototype system.

1. If multiple users from different domains are requiring for the same data, the user with highest *domain_strength* has the priority to use the data while other users (although can view the data) have to wait for him until his session is finished.
2. Multiple users in the same domain can also access to the same data concurrently, but only one user can edit the data under the control of session manager, others can only view the data until the ongoing edit session releases the control.

When the ongoing session is finished and data is checked in, database will trigger a program automatically to inform other views to reload the model so that EPM will be consistent.

6.3.4.2 Security Control

To prevent unauthorized access of product model data, a security manager is used to control the data security, see figure 6.1. This is carried out based on the following security mechanisms.

- ***Database Username and Password***

To access the database, a user must have a valid username of the database. Each username has an associated password to prevent unauthorized use.

- ***User privilege control***

A privilege is a right to run a particular type of SQL statement. Some examples of privileges include the right to:

- (1) Connect to the database (create a session)
- (2) Create a table in your schema
- (3) Select rows from someone else's table
- (4) Execute someone else's stored procedure

The privileges of an Oracle database can be divided into two categories: system privileges and schema object privileges.

- ***System Privileges***

System privileges allow users to perform a particular system-wide action or a particular action on a particular type of schema object. For example, the privileges to create a table

space or to delete the rows of any table in the database are system privileges. Many system privileges are available only to administrators and application developers because the privileges are very powerful.

□ *Schema Object Privileges*

Schema object privileges allow users to perform a particular action on a specific schema object. For example, the privilege to delete rows of a specific table is an object privilege. Object privileges are granted (assigned) to users so that they can use a database application to accomplish specific tasks.

Privileges are granted to users so that users can access and modify data in the database. A user can receive a privilege in two different ways:

- (1) Privileges are granted to users explicitly. For example, the privilege to insert records into the employees' table can be explicitly granted to the user SCOTT.
- (2) Privileges are granted to roles (a named group of privileges), and then the role can be granted to one or more users. For example, the privilege to insert records into the employee's table can be granted to the role named CLERK, which in turn can be granted to the users SCOTT and BRIAN.

• *User Role Control*

Because roles enable easier and better management of privileges, privileges are normally granted to roles and not to specific users. In this section we will explain roles and their use.

Roles are named groups of related privileges that can be granted to users, for example, *connect*, *resource* and *DBA* (Database Administrator) role. Roles are designed to ease the administration of end-user system and schema object privileges. The DBA can create a role with a password to prevent unauthorized use of the privileges granted to the role.

Typically, an application is designed so that when it starts, it enables the proper role. As a result, an application user does not need to know the password for an application's role.

Database roles have the following functionality:

- (1) A role can be granted system or schema object privileges.
- (2) A role can be granted to other roles. However, a role cannot be granted to itself and cannot be granted circularly. For example, role A cannot be granted to role B if role B has previously been granted to role A.
- (3) Any role can be granted to any database user.
- (4) Each role granted to a user is, at a given time, either enabled or disabled. A user's security domain includes the privileges of all roles currently enabled for the user and excludes the privileges of any roles currently disabled for the user. Oracle allows database applications and users to enable and disable roles to provide selective availability of privileges.
- (5) An indirectly granted role is a role granted to a role. It can be explicitly enabled or disabled for a user. However, by enabling a role that contains other roles, you implicitly enable all indirectly granted roles of the directly granted role.

On the basis of above described security maintaining mechanisms, users are separated into several groups. Each group has different access right, which is controlled directly by database manager using database roles. By defining different roles, different user groups may have different access right, e.g. *select only* for certain tables, *select and update* for certain tables, *all privilege* to all tables. All this management data are stored in the database. In this way, data security of product data can be achieved.

6.3.5 Web-based Multi-client Collaboration

In the prototype system, collaboration among multiple clients can be supported by the collaboration manager in the server side. When several clients are participating in a product design session collaboratively, all the clients should first register in the session manager after security checking, to specify product related information and domain information. This is illustrated in figure 6.10. Two design views and a manufacturing view participated in the design of *fix_jaw*, a part which belongs to the product vise. After registration through the session manager, the collaboration can be started.

In design view, the part *fix_jaw* constains seven features which are *extrusion*, *wedge_cut*, *slot*, two *counter_bore_holes* and two *simple_holes*, as shown in figure 6.11. The constraint graph of the *fix_jaw* is shown in figure 6.7.

Figure 6.10 Registration of a collaboration session with the session manager

Figure 6.11 Design features of *fix_jaw*

During collaboration, only one edit view is allowed. In the edit view, after each modeling operation, it will send feature operation instead of entire file to the server side. Figure 6.12 shows sending feature operation from the edit view after adding two *simple_hole* features. In this case, the size of the feature operation file is only 4.07 KB, while the size of the whole part file is 28.4 KB. In real applications, part files can be huge depending on the complexity of geometry. Clearly, by sending feature operations, the communication load between server and distributed clients can be minimized. The feature operation in this case is shown below:

```
feature_operation:
feature name: SIMPLE_HOLE(0)
operation_type: add feature
feature parm:
  diameter : 3.75
  depth : derived
  Taper_angel : 0
  through_flag: 0
feature_direction
  dir_x[0]=1.000000
  dir_y[1]=0.000000
```

```

    dir_z[2]=0.000000
location:
    coplanar_constraint(5)
    coplanar_constraint(6)
    distance_constraint(7)
    distance_constraint(8)
#
Constraint name: coplanar_constraint(5)
feature name: SIMPLE_HOLE(0)
constraint_entity: hole_start
ref_entity: FACE_extrusion(0)_f0
#
Constraint name: coplanar_constraint(6)
feature name: SIMPLE_HOLE(0)
constraint_entity: hole_end
ref_entity: FACE_extrusion(0)_f3
#
constraint name: distance_constraint(7)
feature name: SIMPLE_HOLE(0)
constraint_entity: hole_center
ref_entity: FACE_extrusion(0)_f5
distance: 8.13
#
constraint name: distance_constraint(8)
feature name: SIMPLE_HOLE(0)
constraint_entity: hole_center
ref_entity: FACE_extrusion(0)_f1
distance: 8.13
#
...

```

Upon receiving a feature operation, product manager in the server side will re-evaluate the feature model using different application packages and keep a consistent runtime product model in the server side. According to the domain of the participants, different server actions will be executed. For those who have the same domain with current edit client, all feature operations will be send through data network to the server side; then, the server side will validate the feature model after which the feature operation will be sent to the corresponding views.

Figure 6.12 Send feature operation from the edit view client

Figure 6.13 Receive feature operation and update the model in another design view

In the *fix_jaw* design example, after receiving the feature operation, the server side will create the two *simple_holes* according to their constraints and parameters on the basis of history-independent model evaluation approach and constraint solving mechanisms; then send the operation to another design view; finally, this view can evaluate and update the feature model. The result after model evaluation is shown in figure 6.13. For those who have the different domain with current client, the product manager will first call the corresponding application packages to re-evaluate the view. This kind of model evaluation contains two actions. If only *modify feature operation* type exists, the corresponding application view packages will try to automatically propagate the changes by solving inter-application constraints and local constraints of the corresponding view. If automatic model re-evaluation is not successful due to constraint conflicts (or the view does not exist), the product manager will require the client to open the both views in the client side by downloading the entire file from the server side. Figure 6.14 shows the result with the loaded design model together with the manufacturing view, where two *simple_hole* features were added in the design view. The interactive resolving process is carried out as follows: firstly, create the stock automatically or manually by drawing on the basis of design feature model; then, interactively identify manufacturing features by predefined feature shape and choosing the machining tool as well as cutting tool; finally, generate manufacturing feature model. In such a way, collaboration among distributed clients can be realized.

6.3.6 Web-based Access

The prototype system also supports online communication between distributed clients and database server through the web. For this purpose, all the clients should have the access to Intranet/Internet. Each client can check in and check out data through the MDAI controlled by collaboration manager.

Figure 6.14 The manufacturing features generated

In the prototyped the system, the MDAI is established adopting ASP (active server page) technology. Apache is used as the “HTTP” server to provide data communication service through the web. A security manager in the web server side is responsible for preventing unauthorized access. Figure 6.15 shows a client access the MDAI by providing *username* and *password*.

Once the user has been successfully logged in after the security checking, the user can check in the product information, this procedure is illustrated in figure 6.16. By specifying product related information, the 3D model can be uploaded to the server side. Then, on the server side, product manager will validate the model by different application view packages and finally check the information into the database. The mechanism for data check in has been addressed in detail in section 6.3.3.1.

Figure 6.15 MDAI security management

Figure 6.16 Product information check-in through MDAI

Figure 6.17 Product information check-out through MDAI

If the user wants to check out the product information from the database, MDAI also provides database *search* and data *check out* functions. By specifying the search items, e.g. the product name, the *search* function will search in the database, after which the matching product with all its containing parts will be checked out. The mechanism for data check-out has been described in section 6.3.3.2. Finally, the user can download the file to the client side for further processing. This procedure is illustrated in figure 6.17.

6.4 Summary

In this chapter, an overall design for web-based, database-driven, collaborative feature modeling is proposed. It embodied the feature representation schema definitions under the unified feature modeling framework, feature-oriented database creation on the basis of mapping mechanisms and feature model re-evaluation and validation. Moreover, in response to the trend of collaborative product development, the proposed system is realized

with the web technology. By doing so, it is able to expedite information sharing among multiple, distributed clients based on feature-oriented database. The proposed system includes several servers, namely, web server, application object server, feature object server and database server. Functions of all these modules are explained. In order to support low-level product and process information, a geometric modeler is tightly integrated with the feature-oriented database to provide modeling service. Furthermore, *save()* and *restore()* algorithms for data management in feature-oriented database are described followed by the mechanisms to control data concurrency and data security. Finally, the prototype system is described to illustrate the capabilities of the approach and proposed in this research. The proposed system has implemented the concepts developed in chapters 3, 4 and 5. The prototyped system is able to support some simple online collaboration among distributed product development team members. Ideally, this could provide an effective way to shorten the lead-time and reduce labor cost for product development.

CHAPTER 7 CONCLUSIONS & FUTURE WORK

7.1 Conclusions

This research investigates the mechanisms and implementation techniques of a feature-oriented database for web-based collaborative product development.

Ideally, a collaborative engineering environment enables flexible information sharing among engineers involved in product development who are usually geographically distributed. A common, shared and consistent product model is crucial for information sharing. Compared with traditional file-based information sharing, a database is generic and scalable for reading, writing, updating and deleting operations and therefore is suitable for information integration. However, from the literature review, it appears that the following topics have not been very well handled in the past:

- *A suitable information integration infrastructure*
- *Information loss*
- *Feature semantic problems*
- *Duplicated data and potential conflicts*

The research work presented in this thesis is devoted to bridge these gaps via feature-oriented database, and in turn, to support information modeling and sharing for collaborative product development. Accordingly, the proposed feature-oriented database framework is developed under a four-layer information integration infrastructure. The novelty of this research is a generic feature model, on the basis of which application feature sub-models can be defined; and EPM (entire product model) can be consistently managed. This generic feature model gives a well-defined feature base class. In addition, a history-independent feature evaluation approach, which is based on feature dependency graph, is developed which can dynamically maintain the runtime feature model in

database, and theoretically reduce computation cost. Hence, it can further maintain feature semantics explicitly.

More specifically, a mapping mechanism from original EXPRESS-defined feature model to the target database schema has been developed, in which geometrical representation schema and generic feature representation schema in the database are specified. Furthermore, in order to provide lower-level geometrical modeling functions in the feature model, a geometrical modeler is incorporated; then another mapping mechanism, from EXPRESS-defined geometric model to ACIS workform format has been implemented. Through case study, it can be concluded that by applying the proposed approach and using the mechanisms developed in this work, a feature-oriented database can be achieved. Global collaborative product development is feasible with the aid of feature-oriented database.

7.2 Contributions

The contributions stemmed from this work include:

1. *A proposed information integration infrastructure.* A four-layer information integration infrastructure [Tang 2004] has been proposed to realize information integration, which is crucial for information sharing among product development team members in a global collaborative environment. The four layers are application, information, representation and physical layers. Within this infrastructure, EPM describes low-level geometrical and non-geometrical information across applications, and manages the associative constraints and the domain entity references. The high-level feature objects are organized by different sub-models in the application feature layer. Application feature sub-models define specific views of the EPM for different applications. The unified feature model

provides a generic feature modeling layer with all the common definitions for different applications. Each application defines its feature model on the basis of the unified feature model. It is essential that each feature type has a well-defined semantics. Different types of constraints are used for capturing feature semantics in the context of product shape models. All the models, e.g. application feature model and unified feature model, are represented in EXPRESS language. For implementation, such EXPRESS-defined information models have been mapped to database schema (for data storage), and programming data structure (i.e. ACIS workform format). A database development framework in the physical layer has been specified in order to provide the repository for all kinds of information, e.g. geometric and non-geometric entities, feature entities and others.

2. *A generic feature model.* Under the unified feature modeling framework, a generic feature model is described for different applications, which include feature shape representation with parameters and constraints as well as all the relevant functional methods such that all feature entities can be handled.

To represent the shape of a feature, the feature geometry and topology with parameters, and the associative constraints are specified in detail. Their manipulation functions include creation, modification and deletion functions. At the same time, feature parameters can provide user interfaces to create and modify features in the modeling operations.

In the generic feature model, constraints are defined to specify relations among features, geometric and topological entities, and to provide invariant characteristics in the model. They are implemented as attributes attached to features and other type of topological entities.

3. *Integration of a geometrical modeler and the feature-oriented database for supporting low-level geometrical functionalities.* It can be appreciated that database allows the handling of a large volume of data and is generic for reading, writing, updating and deleting operations; and the DBMS (database management system) can ensure the security and transparency for the users of product data. Therefore databases are appropriate for information sharing among product development team members. However, so far, no database has the capability to handle engineering-related, fine-grain geometrical data (as well as features) due to the complexity of definitions, associative relations, and dynamic changes occurred in the course of applications.

A geometrical modeler (ACIS) has been tightly integrated into the proposed system to provide lower-level geometrical modeling services on the basis of which higher-lever feature modeling service can be realized. Rules of mapping from the generic feature model to the workform (ACIS) format as well as to target database schemas have been investigated in details. Based on the mapping mechanisms, a generic feature representation schema with modeler-supported entities is given. Then feature representation schema in database has also been developed. It can be envisaged that this integration can support the proposed feature-oriented database with powerful geometrical manipulation functions, and at the same time, enables the flexibility and transparency for CAx users and ensures the data consistency, concurrency and security.

4. *A history-independent approach for feature model evaluation.* Maintaining feature semantics and reduce computational cost are the key research topics for feature-based design. The approach proposed in this work is to maintain the feature precedence order by using a feature dependency graph and cellular model's

overlapping detection function. Rules and algorithms for the precedence order determination among features are described. Mechanisms for history-independent feature model evaluation is proposed and analyzed in order to minimize the computational cost and solve semantic problems suffered by most current CAD systems.

5. *System architecture for a web-based collaborative feature modeling system.*

According to the identified system requirements, the client-server system architecture [Tang 2004a] has been proposed to enable information sharing among CAX applications. The proposed system allows multi-view data accesses via the security and session managers. Methods for controlling the concurrency and, at the same time, maintaining the data security are suggested. The application object server is designed to provide different application interfaces (e.g. design and process planning views) for different users such that users can interactively carry out feature-based operations in their individual context. The database server provides physical storage for all kinds of data, either to be persistently maintained or to be shared among different applications. By using feature operations, communication load among multiple clients and database server can be greatly reduced.

7.3 Limitations

This thesis presents a novel solution for developing a feature-oriented database to support collaborative product development in a global distributed environment. The limitations of the prototype system include:

- The application of the proposed feature-oriented database is mainly targeted to mechanical product design. There is great potential of applying the current work to other areas (e.g. civil engineering), but further research is needed.
- Currently, feature representation schema for the proposed system supports a few typical design and machining features. To further support other feature models for different views, more development works are needed.
- The proposed system currently does not incorporate mechanisms to support knowledge-based product development. More research work is needed to build up the connections between this feature database and knowledge facts as well as the inference engine.

7.4 Future Work

In response to the limitations presented in the previous section and to further enhance the methodologies and approaches proposed in this work, some future works are suggested as follows:

- To broaden the application scope of the proposed methodology, more comprehensive application feature models should be defined and the corresponding feature representation schema in database should be further enhanced to support more application views.
- As the proposed collaboration system is web-oriented, a data format for internet communication is important. XML is more suitable because it is platform independent, web friendly, extensible and self-validated. Therefore, XML should be applied in the proposed system for interactions and communication.
- More research can be focused on the strategies for fine-grain data security management. For example, some of the users should not be given permanent access

rights to the product data because their involvement is time-limited and based on “need-to-know” arrangement. Therefore, the strategies for dynamically maintaining the data security should be investigated.

- A knowledge model should be added on the basis of the current feature model so that the feature-oriented database can be extended to interface with knowledge bases. This can be realized by defining database methods to support knowledge facts extraction, and other common knowledge based processes, such as searching, matching, indexing, reasoning and sharing during product development.

REFERENCES

- [ACIS 2001] ACIS Online Help User's Guide. Available at: <http://www.spatial.com>. (Accessed in September 2005)
- [ACS Software] ACS software. <http://www.acssoftware.com/>. (Accessed in September 2005)
- [An 1995] An, D., Leep, H. R., Parsaei, H. R. and Nyaluke, A. P., "A Product Data Exchange Integration Structure Using PDES/STEP for Automated Manufacturing Applications", *Computers & Industrial Engineering*, Vol. 29, No. 1-4, pp. 711-715, 1995.
- [Atkinson 1989] Atkinson, M., Bancilhon, F., Dewitt, D., Dittrich, K., Maier, D. and Zdonik, S., "The Object-Oriented Database System Manifesto", *Proceedings of 1st International Conference on Deductive and Object-Oriented Databases*, Kyoto, Japan, 1989.
- [AutoEDMS 2001] *AutoEDMS Whitepaper*, "The AutoEDMS Document Management & Workflow System", 2001.
- [Bettig 2001] Bettig, B. and Shah, J. J., "Derivation of a standard set of geometric constraints for parametric modeling and data exchange", *Computer-Aided Design*, Vol. 33, pp. 17-33, 2001.
- [Bhandarkar 2000] Bhandarkar, M. P., "STEP-based Feature Extraction From STEP Geometry for Agile Manufacturing", *Computer in Industry*, Vol. 41, pp. 3-24, 2000.
- [Bidarra 1998] Bidarra, R., de Kraker, K. J. and Bronsvoort, W. F., "Representation and management of feature information in a cellular model", *Computer-Aided Design*, Vol. 30, pp. 301-313, 1998.
- [Bidarra 1999] Bidarra, R. and Bronsvoort, W. F., "History-independent boundary evaluation for feature modeling", *Proceedings of DETC'99 1999 ASME Design Engineering Technical Conferences*, Las Vegas, 1999.

- [Bidarra 2000] Bidarra, R. and Bronsvort, W. F., "Semantic feature modeling", *Computer-Aided Design*, Vol. 32, pp. 201–225, 2000.
- [Bidarra 2001] Bidarra, R., van den Berg, E. and Bronsvort, W. F., "Collaborative modeling with features", *Proceedings of DETC'01 2001 ASME Design Engineering Technical Conferences*, Pittsburgh, Pennsylvania, 2001.
- [Bidarra 2002] Bidarra, R. and Bronsvort, W. F., "Persistent naming through persistent entities", *Proceedings of Geometric Modeling and Processing 2002 - Theory and Applications, 10-12 July, Wako, Japan*, Suzuki H and Martin R (eds). IEEE Computer Society, Los Alamitos, pp. 233-240, 2002.
- [Bidarra 2005] Bidarra, R., Madeira, J., Neels, W. J. and Bronsvort, W. F., "Efficiency of boundary evaluation for a cellular model", *Computer-Aided Design*, Vol. 37, pp. 1266-1284, 2005.
- [Borning 1996a] Borning, A., Anderson, R. and Benson, B. F., "Indigo: a local propagation algorithm for inequality constraints", *Proceedings of the 1996 ACM Symposium on User Interface Software and Technology*, pp.129-136, 1996.
- [Borning 1996b] Borning, A., Anderson, R. and Benson, B. F., "The Indigo algorithm", *Technical Report 96-05-01*, Department of Computer Science and Engineering, University of Washington, USA, 1996.
- [Borning 1998] Borning, A. and Benson, B. F., "Ultraviolet: a constraint satisfaction algorithm for interactive graphics", *CONSTRAINTS: An International Journal, Special Issue on Constraints, Graphics, and Visualization*, Vol. 3, pp.1-26, 1998.
- [Bray 1997a] Bray, T., Paoli, J. and Sperberg-McQueen, C. M., *Extensible Markup Language (XML): Part I. Syntax*, World Wide Web Consortium Working Draft (Work in Progress), August 1997.

- [Bray 1997b] Bray, T. and DeRose, S., *Extensible Markup Language (XML): Part II. Linking*, World Wide Web Consortium Working Draft (Work in Progress), July 1997.
- [Bray 1998] Bray, T. and Paoli, J., *Extensible Markup Language (XML) 1.0*, W3C Recommendation, Sperberg-McQueen CM, 1998.
- [Bronsvoort 1997] Bronsvoort, W. F., Bidarra, R. and Dohmen, M., "Multi-view feature modeling and conversion", In: *Geometric Modeling: Theory and Practice - The State of the Art*, ed. Strasser W, Klein R, and Rau R, Springer, Berlin, pp.159-174, 1997.
- [Bronsvoort 2001] Bronsvoort, W. F., Bidarra, R. and Noort, A., "Semantic and multiple-view feature modeling: towards more meaningful product modeling". In: *Geometric Modeling - Theoretical and Computational Basis towards Advanced CAD Applications*, ed. F. Kimura, Kluwer Academic Publishers, Dordrecht, pp. 69-84, 2001.
- [Bronsvoort 2004] Bronsvoort, W. F. and Noort, A., "Multiple-view feature modeling for integral product development", *Computer-Aided Design*, Vol. 36, pp. 929-946, 2004.
- [Bryant 1996] Bryant, R. V. E., "Common Product/ Process Models for Interfacing Manufacturing Simulation, Process Planning and CAD", *Proceedings of the 1996 ASME Conference*, Irvine,
- [Burton 1998] Burton, P., "Technologies & Standards", *Technical Report from the European Commission supported WIDEBEAM project*. http://www.isomatic.co.uk/WBStandards.htm#Peter_Burton. Last access May, 2005.
- [CAD/CAM-E] CAD/CAM-E website. <http://www.cadcam-e.com/> (Accessed in September 2005)
- [Capoyleas 1994] Capoyleas, V., Chen, X., and Hoffmann, C. M., Generic naming in generative, constraint-based design, *Technical Report 94-011*, Purdue University, Computer Science, 1994.

- [Capoyleas 1996] Capoyleas, V., Chen, X. and Hoffmann, C. M., "Generic naming in generative, constraint-based design", *Computer-Aided Design*, Vol. 28, No. 1, pp.17-26, 1996.
- [Carter 1992] Carter, D. E., and Baker, B. S., *Concurrent Engineering: The Product Development Environment for the 1990s*, Addison Wesley, 1992.
- [Chan 2000] Chan, F. T. S., Zhang, J., Lau, H. C. W. and Ning, A., "Information Integration Platform for CIMS", *Proceedings of the 2000 IEEE International Conference*, Vol. 2, pp. 743-747, 2000.
- [Chang 1998] Chang, T. C., Wysk, R. A., and Wang, H. P., *Computer-aided manufacturing (Second Edition)*, Prentice Hall, 1998.
- [Chao 2001] Chao, P. Y., and Wang, Y., "A Data Exchange Framework for Networked CAD/CAM", *Computer in Industry*, Vol. 44, pp. 131-140, 2001.
- [Chen 1994] Chen, X., and Hoffmann, C. M., On editability of feature-based design. *Technical Report CSD 94- 067*, Purdue University, Computer Science, 1994.
- [Chen 1996] Chen, X. and Hoffmann, C. M., "On Editability of Feature-based Design", *Computer-Aided Design*, Vol. 27, No.12, pp. 905-914, 1996.
- [Chen 2004a] Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Unified feature modeling scheme for the integration of CAD and CAX", *Computer-Aided Design & Applications*, Vol. 1, Nos. 1-4, pp. 595-602, 2004.
- [Chen 2004b] Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Unified feature based integration of design and process planning", *Proceedings of the 34th International MATADOR conference*, pp. 63-68, 2004.
- [Chen 2005a] Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., "Knowledge-Based Reasoning in a Unified Feature Modeling Scheme", *Computer-Aided Design & Applications*, Vol. 2, Nos. 1-4, CAD'05, pp. 173-182, 2005.

- [Chen 2005b] Chen, G., Ma, Y. S., Ming, X. G., Thimm, G., Lee, S. S. G., Khoo, L. P., Tang, S. H., and Lu, W. F., "A unified feature modeling scheme for multi-applications in PLM", *proceedings of The 12th ISPE International Conference on Concurrent Engineering (CE2005): Research and Applications -Next Generation Concurrent Engineering: Smart and Concurrent Integration of Product Data, Services, and Control Strategies*, M. Sobolewski and P. Ghodous (Eds.), ISPE, Dallas, pp. 343-348, 2005.
- [Chen 2005] Chen, J. Y., Ma, Y. S., Wang, C. L. and Au, C. K., "Collaborative Design Environment with Multiple CAD Systems", *Computer-Aided Design & Applications*, Vol. 2, Nos. 1-4, pp. 367-376, 2005.
- [Clark 1995] Clark, A. L. and Staley, S. M., "STEP AP203 Data Exchange Study", *Proceedings of the Symposium on Solid Modeling and Applications*, p 213-224, 1995.
- [Clark 1987] Clark, K., Chew, B. and Fujimoto, T., "Product Development in the World Auto Industry", *Brookings Papers on Economic Activity*, pp. 729-771, March 1987.
- [Clark 1990] Clark, S., "An Introduction to the NIST PDES Toolkit," *Technical Report NISTIR 4336*, National Institute of Standards and Technology, Gaithersburg, Maryland, May 1990.
- [Cunningham 1988] Cunningham, J. J. and Dixon, J. R., "Designing with features: the origin of features", *Proceedings of ASME Computers in Engineering Conference*, San Francisco, pp. 237-243, 1988.
- [Cutkosky 1996] Cutkosky, M. R., "MADEFast: Collaborative Engineering Over the Internet", *Communications of the ACM*, Vol. 39, No. 9, pp. 78-87, 1996.
- [Da Cunha 2002] da Cunha, R. R. M. and Dias, A., "Feature-based database evolution approach in the design process", *Robotics and Computer Integrated Manufacturing*, Vol. 18, pp. 275-281, 2002.

- [Date 1998] Date, C. J. and Darwen, H., *Foundation for object/relational databases: the third manifesto*, Addison-Wesley, 1998.
- [Date 2000] Date, C. J. and Darwen, H., *Foundation for future database systems: the third manifesto: a detailed study of the impact of type theory on the relational model of data, including a comprehensive model of type inheritance*, Addison-Wesley, 2000.
- [Dereli 2002] Dereli, T. and Filiz, H., "A note on the use of STEP for interfacing design to process planning", *Computer-Aided Design*, Vol. 34, pp. 1075-1085, 2002.
- [Dohmen 1998] Dohmen, M., "Constraint-based feature validation", *PhD thesis*, Delft University of Technology, Netherlands, 1998.
- [Domazet 2000] Domazet, D. S., Miao, C. Y., Calvin, C. F. Y., Kong, H. P. H. and Goh, A., "An Infrastructure for Inter-Organizational Collaborative Product Development", *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.
- [Dixon 1990] Dixon, J. R. and Duffy, M. R., "The neglect of engineering design", *California Management Review*, Vol. 32, pp. 9-23, 1990.
- [Dumaine 1989] Dumaine, B., "How Manager Can Succeed Through Speed", *Fortune*, Vol. 119, pp. 54-59, 1989.
- [El-Mehalawi 2003a] El-Mehalawi, M. and Miller, R. A., "A database system of mechanical components based on geometric and topological similarity. Part I: representation", *Computer-Aided Design*, Vol. 35, pp. 83-94, 2003.
- [El-Mehalawi 2003b] El-Mehalawi, M. and Miller, R. A., "A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment", *Computer-Aided Design*, Vol. 35, pp. 95-105, 2003.
- [Emmel 2000] Emmel, J., "OneSpace Integrating Collaboration Technology and Enterprise PDM", *Technical Whitepaper of CoCreate Software GmbH*, 2000.

- [Everitt 1988] Everitt, P. B., Product data exchange specification: RAMP's unique implementation of the PDES specification, *Society of Manufacturing Engineers*, 1988.
- [Finin 1992] Finin, T., Mckay, D. and Fritzson, R., "An Overview of KQML: A Knowledge Query and Manipulation Language", *Technical Report*, University of Maryland, College Park, MD, 1992.
- [Finin 1993] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., Mckay, D., Mcguire, J., Pelavin, R., Shapiro, S. and Beck, C., "Specification of the KQML Agent-communication Language", *The DARPA Knowledge Sharing Initiative External Interface Working Group*, 1993.
- [Fortier 1997] Fortier, P. J., *Database systems handbook*, New York: McGraw-Hill, 1997.
- [Fu 2003] Fu, M. W., Ong, S. K., Lu, W. F., Lee, I. B. H. and Nee, A. Y. C., "An approach to identify design and manufacturing features from a data exchanged part model", *Computer-Aided Design*, Vol. 35, pp. 979–993, 2003.
- [Fudos 1997] Fudos, I. and Hoffmann, C. M., "A Graph-Constructive Approach to Solving Systems of Geometric Constraints", *ACM Transactions on Graphics*, Vol. 16, No. 2, pp. 179-216, 1997.
- [Fynes 2005] Fynes, B. and Burca, S. D., "The effects of design quality on quality performance", *International Journal of Production Economics* Vol. 96, pp. 1-14, 2005.
- [Gao 2004a] Gao, J., Zheng, D.T., and Gindy, N., "Extraction of machining features for CAD/CAM integration", *International Journal of Advanced Manufacturing Technology*, Vol. 24, pp. 573–581, 2004.
- [Gao 2004b] Gao, J., Zheng, D.T., and Gindy, N., "Mathematical representation of feature conversion for CAD/CAM integration", *Robotics and Computer-Integrated Manufacturing*, Vol. 20, pp. 457-467, 2004.

- [Geelink 1995] Geelink, R., Salomons, O. W., Van Slooten, F., Van Houten, F. J. A. M. and Kals, H. J. J., "Unified feature definition for feature based design and feature based manufacturing", *Proceedings of the 15th Annual International Computers in Engineering Conference the 9th Annual ASME Engineering Database Symposium*, pp. 517-533, 1995.
- [Gomes 1993] Gomes, A., Bidarra, R. and Teixeira, J. C., "A cellular approach for feature-based modeling", In: *Graphics modeling and visualization in science and technology*, ed. Gobel, M. and Teixeira, J. C., pp. 128-143, Springer-Verlag, Heidelberg, 1993.
- [Gonzalez 2002] Gonzalez, F. J. M. and Palacios, T. M. B., "The effect of new product development techniques on new product success in Spanish firms", *Industrial Marketing Management*, Vol. 31, pp. 261-271, 2002.
- [Green 1997] Green, R., In: *CADENCE CHANNEL*, "Key kernels: ACIS and Parasolid", 1997. Available at: <http://cadence.advanstar.com/1998/1298/supgreen.html>. (Accessed in September 2005)
- [Griffin 1995] Griffin, C. R. and Patel, P., "*The Transfer and Exchange of Engineering Information*", Printed and published by the IEE, Savoy Place, London WC2R 0BL, UK, 1995.
- [Hardwick 1991] Hardwick, M., "Managing Change Using STEP/EXPRESS", *Technical Report 91016*, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1991.
- [Hardwick 1994] Hardwick, M., "Towards Integrated Product Databases Using Views", *Technical Report 94003*, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1994.
- [Hoffmann 1998] Hoffmann, C. M. and Arinyo, R. J., "CAD and the product master model", *Computer-Aided Design*, Vol. 30, No. 11, pp. 905-918, 1998.

[Hoffmann 2002] Hoffmann, C. M. and Arinyo, R. J., *Handbook of computer-aided geometric design*, chapter parametric modeling: pp. 519-542, Amsterdam: North-Holland, 2002.

[Holland 2002] Holland, P., Standring, P. M., Long, H. and Mynors, D. J., "Feature extraction from STEP (ISO10303) CAD drawing files for metal-forming process selection in an integrated design system", *Journal of Materials Processing Technology*, Vol. 125-126, pp. 446-455, 9 September 2002.

[Huang 2001] Huang, G. Q. and Mak, K. L., "Web-integrated manufacturing: recent developments and emerging issues", *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 1, pp. 1-2, 2001.

[ISO 1994a] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 1: Overview and Fundamental Principles, *ISO 10303-1:1994* (E), ISO, Geneva, 1994.

[ISO 1994b] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 11: Description Methods: The EXPRESS Language Reference Manual, *ISO 10303-11:1994* (E), ISO, Geneva, 1994.

[ISO 1994c] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 21: Implementation Methods: Clear Text En-coding of the Exchange Structure, *ISO 10303-21:1994* (E), ISO, Geneva, 1994.

[ISO 1994d] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support, *ISO 10303-41:1994* (E), ISO, Geneva, 1994.

[ISO 1994e] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 42: Integrated Generic Resources: Geometric and Topological Representation, *ISO 10303-42:1994* (E), ISO, Geneva, 1994.

[ISO 1994f] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 203: Application Protocol: Configuration Controlled Design, *ISO 10303-203:1994* (E), ISO, Geneva, 1994.

[ISO 1995a] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 22: STEP Data Access Interface, *ISO Document TC184/SC4 WG7 N392*, July 1995.

[ISO 1995b] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 23: C++ Language Binding to the Standard Data Access Interface Specification, *ISO Document TC184/SC4 WG7 N393*, July 1995.

[ISO 1995c] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 27: Standard Data Access Interface — C Language Late Binding, *ISO Document TC184/SC4 WG7 N394*, July 1995.

[ISO 2000a] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 24: Java™ programming language binding to the standard data access interface with Internet/Intranet extensions, *ISO Document ISO TC 184/SC4/WG11 N060*, June 2000.

[ISO 2000b] Industrial Automation Systems and Integration — Product Data Representation and Exchange — Part 224: Mechanical product definition for process planning using machining features, *ISO Document ISO TC 184/SC4/WG3 N854*, Dec. 2000.

[Jasnoch 1996] Jasnoch, U. and Haas, S., “A collaborative Environment Based on Distributed Object-Oriented Databases”, *Computer in Industry*, Vol. 29, pp. 51-61, 1996.

[Jha 2000] Jha, K. and Gurumoorthy, B., “Automatic propagation of feature modification across domains”, *Computer-Aided Design*, Vol. 32, pp. 691-706, 2000.

- [Jiao 1999] Jiao, J. X., "A Requirement Management Database System for Product Definition", *Integrated Manufacturing Systems*, Vol. 10, pp.146-153, 1999.
- [Joe 2000] Joe, Y. S., Svetan, M. R. and George, V., "Distributed collaborative Design and Manufacturability Assessment for Extended Enterprise in XML-based Agent system", *Enabling Technologies: Infrastructure for Collaborative Enterprises, Proceedings of IEEE 9th International Workshops on*, pp. 260-265, 2000.
- [Joshi 1988] Joshi, S. B. and Chang, T.C., "Graph-based heuristics for recognition of machined features from a 3D solid model", *Computer-Aided Design*, Vol. 20, pp. 58-66, 1988.
- [Juri 1990] Juri, A.H.; Saia, A.; De Pennington, A., , "Reasoning About Machining Operations Using Feature-Based Models", *International Journal of Production Research*, Vol. 28, No. 1, pp. 153-171, 1990.
- [Kang 1997] Kang, S. H., Kim, N., Kim, C.Y., Kim, Y. and O'Grady, P., "Collaborative Design Using the World Wide Web", *Technical Report*, Dept. Industrial Engineering, Seoul National University, Korea 1997.
- [Kent 1990] Kent, R., Harrod, D. and Conroy, W., *The Initial Graphics Exchange Specification (IGES) version 5.0*, CAD-CAM Data Exchange Technical Center, 1990.
- [Kim 1998] Kim, C. Y., Kim, N., Kim, Y., Kang, S. H. and O'Grady, P., "Distributed Concurrent Engineering: Internet-based Interactive 3-D Dynamic Browsing and Markup of STEP Data", *Concurrent Engineering: Research and Application*, Vol. 6, No.1, pp. 53-70, March 1998.
- [Kim 2001] Kim, Y., "A Distributed, Open, Intelligent Product Data Management System", *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 2, pp. 224-235, 2001.

- [**Kim 2003**] Kim, J. and Han, S., "Encapsulation of geometric functions for ship structural CAD using a STEP database as native storage", *Computer-Aided Design*, Vol. 35, pp. 1161-1170, 2003.
- [**Koparanova 2002**] Koparanova, M. G. and Risch, T., "Complete CAD queried for visualization", *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02)*, IEEE 2002.
- [**Kramer 1992**] Kramer, G. A., "Solving geometric constraints systems: a case study in kinematics", The MIT Press, Cambridge, MA, USA, 1992.
- [**Krebs 1995**] Krebs, T. and Luhrsens, H., "STEP Databases as Integration Platform for Concurrent Engineering", *Proceedings of 2nd International Conference on Concurrent Engineering* (McLean, Virginia, August 23-25 1995), Concurrent Technologies Corporation, Johnstown, PA, pp. 131-142, 1995.
- [**Kung 1999**] Kung, H. K., "Applying Object-oriented Database Technologies in Concurrent Design Processes", *International Journal of Computer Integrated Manufacturing*, Vol. 12, No. 3, pp. 251-264, 1999.
- [**Kumar 2000**] Kumar, A. V. and Yu, L., "Sequential constraint imposition for dimension-driven solid models", *Computer-Aided Design*, Vol. 33, pp. 475-486, 2001.
- [**Kwaiter 1998**] Kwaiter, G., Gaildrat, V. and Caubet, R., "Modeling with constraints: a bibliographical survey", *International Conference on Information Visualization*, 1998.
- [**Labrou 1997**] Labrou, Y. and Finin, T., "A proposal for a new KQML specification", *Technical report: CS-97-03*, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, February 1997.
- [**Lee 1999**] Lee, J. Y., Han, S. B., Kim, H. and Park, S. B., "Network-centric Feature-based Modeling", *Proceedings of Seventh Pacific Conference*, pp. 280-289, 1999.

- [Lee 2003] Lee, K. Y., Kwon, O. H., Lee, J. Y. and Kim, T. W., "A hybrid approach to geometric constraint solving with graph analysis and reduction", *Advances in Engineering Software*, Vol. 34, pp. 103-113, 2003.
- [Li 2002] Li, W. D., Ong, S. K. and Nee, A. Y. C., "Recognizing manufacturing features from a design-by-feature model", *Computer-Aided Design*, Vol. 34, pp. 849-868, 2002.
- [Li 2004] Li, W. D., Ong, S. K., Fuh, J. Y. H., Wong, Y. S., Lu, Y. Q. and Nee, A. Y. C., "Feature-based design in a distributed and collaborative environment", *Computer-Aided Design*, Vol. 36, pp. 775-797, 2004.
- [Li 2002] Li, Y. T., Hu, S. M. and Sun, J. G., "A constructive approach to solving 3-D geometric constraint systems using dependency analysis", *Computer-Aided Design*, Vol. 34, pp. 97-108, 2002.
- [Liang 1998] Liang, W. Y. and Grady, P. O., "An Object-Oriented Formalism for Feature-Based Distributed Concurrent Engineering", *Technical Report TR98-03*, 1998.
- [Light 1982] Light, R. and Gossard, D., "Modification of geometric models through variational geometry", *Computer-Aided Design*, Vol. 14, pp. 209-214, 1982.
- [Loffredo 1998] Loffredo, D., "Efficient Database Implementation of EXPRESS Information Models", *PhD Thesis*, Rensselaer Polytechnic Institute, Troy, New York, May 1998.
- [Ma 2003] Ma, Y. S. and Tong, T., "Associative feature modeling for concurrent engineering integration", *Computers in Industry*, Vol. 51, pp. 51-71, 2003.
- [Ma 2004] Ma, Y. S., Britton, G. A., Tor, S. B., Jin, L. Y., Chen, G. and Tang, S. H., "Design of an feature-object-based mechanical assembly library", *Computer-Aided Design & Applications*, Vol. 1, Nos. 1-4, 2004, CAD'04, pp 379-403.
- [Mahmoud-Jouini 2004] Mahmoud-Jouini, S. B., Midler, C. and Garel, G., "Time-to-market vs. time-to-delivery Managing speed in Engineering, Procurement and

Construction projects”, *International Journal of Project Management*, Vol. 22, pp. 359-367, 2004.

[**Mallick 1992**] Mallick, D. N. and Kouvelis, P., “ Management of Product Design: A strategic Approach”, *Intelligent Design and Manufacturing*, John Wiley & Sons, Inc. pp. 157-177, 1992.

[**Martino 1998**] Martino, T. D., Falcidieno, B. and Habinger, S., “Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment”, *Computer-Aided Design*, Vol. 30, pp. 437-452, 1998.

[**McClatchey 1998**] McClatchey, R., Kovacs, Z., Estrella, F., Le Goff, J. M., Chevenier, G., Baker, N., Lieunard, S., Murray, S., Le Flour, T. and Bazan, A., “The integration of product data and workflow management systems in a large scale engineering database application”, *Proceedings of IDEAS'98: International Database Engineering and Applications Symposium*, pp. 296-302, 1998.

[**Mills 1998**] Mills, A., “*Collaborative engineering and the Internet: linking product development partners via the Web*”, Dearborn, MI: Society of Manufacturing Engineers, 1998.

[**Mittra 1991**] Mittra, S. S., “*Principles of relational database systems*”, Englewood Cliffs, N.J. : Prentice Hall, 1991.

[**Mowbray 1995**] Mowbray, T. J. and Zahavi, R., *The Essential CORBA: Systems Integration Using Distributed Objects*, New York: Wiley, ISBN: 0471106119, 1995.

[**Muller 1993**] Muller, J. and Smith, G., “A Pre-Competitive Project in Intelligent Manufacturing Technology”, *Proceedings of AAAI '93 Workshop on Intelligent Manufacturing Technology*, 1993.

[**Noort 2002**] Noort, A., Hoek, G. F. M. and Bronsvoort, W. F., “Integrating part and assembly modeling”, *Computer-Aided Design*, Vol. 34, pp. 899-912, 2002.

- [Nwagboso 2004] Nwagboso, C., Georgakis, P. and Dyke, D., "Time compression design with decision support for intelligent transport systems deployment", *Computers in Industry*, Vol. 54, pp. 291-306, 2004.
- [OPENDXM] OPENDXM overview. <http://www.prostep.de/en/solutions/opendxm/> (Accessed in September 2005)
- [ORACLE 2004] ORACLE online documentation. Available at: <http://www.oracle.com>. (Accessed in September 2005)
- [Otto 2001] Otto, H. E., "From concepts to consistent object specifications: translation of a domain-oriented feature framework into practice", *Journal of computer science & technology*, Vol. 16, pp. 208-230, 2001.
- [Owen1993] Owen, J., *STEP: An Introduction*, Information Geometers (47 Stockers Avenue, Winchester SO22 5LB, United Kingdom), 1993.
- [Pahng 1998] Pahng, F., "Web-based Collaborative Design Modeling and Decision Support", *Proceedings of ASME Design Engineering Technical Conferences* September 13-16, Atlanta, Georgia, 1998.
- [PDMI2 1999] "PDM status report 09/99", 1999.
- [Pham 1998] Pham, D.V. and Dimov, S.S., "An Approach to Concurrent Engineering", *Journal of Engineering Manufacture*, Vol. 212, pp. 13-27, 1998.
- [Petrie 1996] Petrie, C. J., "Agent-based Engineering, the Web, and Intelligence", *IEEE Expert*, pp. 24-29, December 1996.
- [Pfeifer 1999] Pfeifer, T., "Internet-Intranet-Infranet: a modular integrating architecture", *Proceedings of 7th IEEE Workshop on Future Trends*, pp. 81-86, 1999.
- [Prasad 1996] Prasad, B., *Concurrent Engineering Fundamentals: Integrated Product and Process Organization*, Upper Saddle River, NJ: Prentice Hall, 1996.

- [Pratt 1988] Pratt, M. J., "Synthesis of an optimal approach to form feature modeling", In: *Proceedings of the ASME 1988 Computers in Engineering Conference, Vol. 1*, pp. 263-274, ASME, New York, 1988.
- [Raflik 1990] Raflik, M., *CAD*I Database-An Approach to an Engineering Database*, ECSC-EEC-EAEC, Brussels-Luxembourg, 1990.
- [Raghothama 1998] Raghothama, S. and Shapiro, V., "Boundary representation deformation in parametric solid modeling", *ACM Transactions on Graphics*, Vol. 17, No.4, pp. 259-286, October 1998.
- [Raghothama 2000] Raghothama, S. and Shapiro, V., "Consistent updates in dual representation systems", *Computer-Aided Design*, Vol. 32, pp. 463-477, 2000.
- [Ramakrishnan 2000] Ramakrishnan, R. and Gehrke, J., *Database management systems*, Boston: McGraw-Hill, 2000.
- [Raper 2000] Raper, S. A. and Fukuda, S., "Understanding the collaborative engineering design process in a global environment: observations of Japanese and American approaches", *Proceeding of ICEE'2000*, Taiwan, August 2000.
- [Rappoport 2003] Rappoport, A., "An architecture for universal CAD data exchange", *Proceedings of Solid Modeling '03*, ACM Press, June 2003.
- [Rappoport 2005] Rappoport, A., Steven, S. and Michal, E., "One-dimensional selections for feature-based data exchange", *Proceedings of Solid Modeling '05*, MIT, June 2005.
- [Regli 1997] Regli, W. C. and Gaines, D. M., "A repository for design, process planning and assembly", *Computer-Aided Design*, Vol. 29, pp. 895-905, 1997.
- [Rossignac 1990] Rossignac, J. R. "Issues on feature-based editing and interrogation of solid models", *Computer & Graphics*, Vol. 14, No. 2, pp. 149-172, 1990.
- [Ryan 1995] Ryan, N. and Smith, D., *Database Systems Engineering*, International Thomson Computer Press, 1995.

- [Salas 1997] Salas, A. O. and Rogers, J. L., A Web-based system for monitoring and controlling multidisciplinary design projects, *TM-97-206287*, NASA, 1997.
- [Sannella 1993a] Sannella, M., "The SkyBlue constraint solver", *Technical Report 92-07-02*, Department of Computer Science and Engineering, University of Washington, 1993.
- [Sannella 1993b] Sannella, M., "*The SkyBlue constraint solver and its applications*", First Workshop on Principles and Practice of Constraint Programming, 1993.
- [Sauder 1995] Sauder, D. and Morris, K. C., "Design of a C++ Software Library for Implementing EXPRESS: The NIST STEP Class Library", *Proc. of EUG '95 — The Fifth EXPRESS Users Group Conference*, Grenoble, October 21-22, 1995.
- [Schenck 1994] Schenck, D. and Wilson, P., *Information Modeling the EXPRESS Way*, Oxford University Press, New York, ISBN 0-19-508714-3, 1994.
- [Shah 1988] Shah, J.J. and Rogers, M.T., "Feature Based Modelling Shell: Design and Implementation", *Proceedings of ASME Computers in Engineering Conference*, San Francisco, CA, 1988.
- [Shah 1991] Shah, J. J., "Assessment of features technology ", *Computer-Aided Design*, Vol. 23, No. 5, pp. 331-343, 1991.
- [Shah 1995] Shah, J. J. and Mantyla, M., "*Parametric and feature-based CAD/CAM: concepts, techniques, and applications*", John Wiley and Sons, Inc., 1995.
- [Sheu 1993] Sheu, L. C. and Lin, J. T., "Representation scheme for defining and operating form features", *Computer-Aided Design*, Vol. 25, No. 6, pp. 333-347, 1993.
- [Shin 2000] Shin, Y., Han, S. H. and Bae, D. H., "Integration of homogeneous CAD databases using STEP and Internet", *Decision Support Systems*, Vol. 28, pp. 365-379, 2000.

- [**Smith 1988**] Smith, B. and Rinaudot, G., “Product data exchange specification: first working draft”, Gaithersburg, Md.: National Bureau of Standards, US Dept. of Commerce, 1988.
- [**Smith 1999**] Smith, R. P. and Morrow, J. A., “Product development process modeling”, *Design Studies*, Vol. 20, pp. 237-261, 1999.
- [**Stalk 1988**] Stalk, G., Time- the next source of competitive advantage, *Harvard Business Review*, Vol. 66, pp. 41-51.
- [**Steven 2004**] Steven, S. and Rappoport, A., “Integrated feature-based and geometric CAD data exchange”, *Proceedings of Solid Modeling '04*, ACM Press, Genova, Italy, June 2004.
- [**Suh 1997**] Suh, Y. S. and Wozny, M. J., Interactive feature extraction for a form feature conversion system, In: C. M. Hoffmann, W. F. Bronsvoort, editors, *Solid Modeling '97, Fourth Symposium on Solid Modeling and Applications*, pp. 111-122, New York: ACM Press, 1997.
- [**Tang 2004**] Tang, S. H., Ma, Y. S. and Chen, G., “A feature-oriented database framework for web-based CAx applications”, *Computer-Aided Design & Applications*, Vol. 1, Nos. 1-4, 2004, CAD'04, pp 117-125.
- [**Tang 2004**] Tang, S. H., Ma, Y. S. and Chen, G., “A Web-based Collaborative Feature Modeling System Framework”, *Proceedings of the 34th International MATADOR conference*, pp. 31-36, 2004.
- [**Tang 2005**] Tang, S. H., Ma, Y. S. and Chen, G., “Integration of a Solid Modeler with a Feature-oriented Database”, presented at: CAD'05, 2005.
- [**Terwiesch 1999**] Terwiesch, C. and Loch, C. H., “Measuring the Effectiveness of Overlapping Development Activities”, *Management Science*, Vol. 26, pp. 44-59, 1999.
- [**Thornton 1994a**] Thornton, A. C., “Genetic algorithms versus simulated annealing: satisfaction of large sets of algebraic mechanical design constraints”, In *Artificial*

Intelligence in Design '94, ed. J. S. Gero and F. Sudweeks, Kluwer Academic Publishers, 1994.

[Thornton 1994b] Thornton, A. C., “A Support Tool for Constraint Processes in Embodiment Design”, *ASME Design Theory and Methodology Conference*, 1994.

[Thornton 1996] Thornton, A. C. and Johnson, A. L., “CADET: A Software Support Tool for Constraint Processes in Embodiment Design”, *Research in Engineering Design*, Vol. 8, pp.1-13, 1996.

[Tschrizis 1978] Tschrizis and Klug, “The Ansi/X3/SPARC DBMS Framework: Report of the Study Group on Database Management Systems”, *Information Systems*, 3, 1978.

[Ulrich 1995] Ulrich, K. T. and Eppinger, S. D., *Product Design and Development*, Singapore: McGraw-Hill, 1995.

[Urban 1993] Urban, S. D., “Engineering Data Management: Achieving Integration Through Database Technology”, *Computing & Control Engineering Journal*, 1993.

[Van den Berg 2002] van den Berg, E., Bidarra, R. and Bronsvoort, W. F., “Web-based interaction on feature models”, In: *From Geometric Modeling to Shape Modeling*, Cugini U and Wozny M (eds), Kluwer Academic Publishers, Dordrecht, pp. 99-112, 2001.

[Venkataraman 2001] Venkataraman, S., Shah, J. J. and Summers, J. D., “An investigation of integrating design by features and feature recognition”, *IFIP Conference, FEATS 2001*, Valenciennes, Frances, 2001.

[Wang 2002] Wang, H. F. and Zhang, Y. L., “CAD/CAM integrated system in collaborative development environment”, *Robotics and Computer Integrated Manufacturing*, Vol. 18, pp. 135-145, 2002.

[Wang 2003] Wang, H. F., Zhang, Y. L., Cao, J., Lee, S. K. and Kwong, W. C., “Feature-based collaborative design”, *Journal of Material Processing Technology*, Vol. 139, pp. 613-618, 2003.

- [Wang 2005] Wang, Y. and Nnaji, B. O., “Geometry-based semantic ID for persistent and interoperable reference in feature-based parametric modeling”, *Computer-Aided Design*, Vol. 37, pp.1081-1093, 2005.
- [Wilson 1990] Wilson, P. R., “Information Modeling and PDES/STEP”, *Technical Report 90017*, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1990.
- [Wirth 1976] Wirth, N., *Algorithms + Data Structure = Programs*, Englewood Cliffs, NJ: Prentice-Hall, 1976.
- [Wu 1999] Wu, G. S., Huang, Y., Tseng, S. S. and Zhang, F. Y., “A Knowledge Sharing and Collaboration Information System Model Based on Internet”, IEEE, 1999.
- [Xiao 1996] Xiao, K., “STEP Based CAD/CAM System Integration”, *Tsinghua Science and Technology*, Vol. 1, No. 2, 1996.
- [Xue 2003] Xue, D. and Yang, H., “A concurrent engineering-oriented design database representation model”, *Computer-Aided Design*, Vol. 36, pp. 947-965, 2004.
- [Xue 2004] Xue, D. and Xu, Y., “Web-based distributed system and database modeling for concurrent design”, *Computer-Aided Design*, Vol. 35, pp. 433-452, 2003.
- [Zha 2002] Zha, X. F. and Du, H., “A PDES/STEP-based Model and System for Concurrent Integrated Design and Assembly Planning”, *Computer-Aided Design* Vol. 34, pp. 1087-1110, 2002.
- [Zhang 2000] Zhang, Y.P., “An Internet based STEP Data Exchange Framework for Virtual Enterprises”, *Computers in Industry* Vol. 41, pp. 51-63, 2000.
- [Zheng 1994] Zheng, L. Y., “A STEP-based CAD/CAPP Integration”, *CADDM*, Vol. 4, No. 1, 1994.