Development of collaborative design environment with heterogeneous CAD system.

Chen, Junyong.

2007

Chen, J. (2007). Development of collaborative design environment with heterogeneous CAD system. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/36130>

<https://doi.org/10.32657/10356/36130>

The Development of Collaborative Design Environment with heterogeneous CAD Systems

Chen Junyong

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2007

ABSTRACT

Traditional CAD (Computer-Aided Design) technologies focused on the interactive process between a computer and a single user. This shortcoming greatly constrains distributed collaboration. Collaborative CAD systems have been proposed to facilitate resolution in team design and have gained increasing attention and are considered as the next generation CAD solutions. Ideally, such a collaborative CAD system allows a group of designers, often geographically dispersed, to work seamlessly with one another over a distributed network during a real-time design process. Most of previous research efforts in this area concentrated on incorporating network functions into a typical CAD system in order to enable the mechanisms of collaborative design. The problem is that these research efforts are developed on a specific CAD system, thus restricted.

This research extends the field of collaborative CAD from one specific CAD system to multiple CAD systems and makes efforts to develop a collaborative design environment to support a set of heterogeneous CAD systems such that a product can be co-modeled simultaneously among a team of designers at geographically distributed locations. Four major issues are identified in developing a prototype system. They are: 1) feature-level interoperability; 2) network load and CAD data representation; 3) synchronization and consistency; and 4) concurrency. An *operation*-based solution is proposed to deal with these issues. Within the solution, a novel approach that uses *operation* as the basic transmission data unit to address feature-level interoperability is proposed. Based on the solution, the architecture of a collaborative system is proposed and detailed functional modules in this architecture are explained. An administrator/controller mechanism to improve efficiency and stableness is employed in the proposed system.

The preliminary prototype system is developed based on two commercial CAD systems, Unigraphics and SolidWorks. A nearly real-time collaborative design has been realized.

The experimental results from the prototype system have shown the feasibility of the novel method, which addresses feature-level interoperability and realizes dynamic high-level CAD data sharing/exchange among different CAD systems. Based on an evaluation of the prototype system, future work is then suggested.

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 supervisors, Associate Professor Ma Yongsheng and Assistant Professor Au Chi Kit. They have given me excellent supervision, invaluable advice and continuous attention throughout my research. Under their edifying and enlightening guidance, I have saved a lot of effort from straying around and focused straight on the research issues.

I appreciate all the help, encouragement, advice and cheers from friends in my life, especially from Mr. Tang Shaohui 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 this 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

1. 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.
2. S. H. Tang, Y. -S. Ma, G. Chen and J. Y. Chen, "Design of Feature-oriented Database for Collaborative Product Development", accepted by *International Journal of Advanced Manufacturing Systems*.

TABLE OF CONTENTS

ABSTRACTI

ACKNOWLEDGEMENTSIII

LIST OF PUBLICATIONSIV

TABLE OF CONTENTSV

LIST OF FIGURESIV

LIST OF TABLESXI

LIST OF ABBREVIATIONSXII

CHAPTER 1 INTRODUCTION1

1.1 Background..... 1

1.2 Objectives 6

1.3 Scope of research project 7

1.4 Contributions 7

1.5 Thesis organization 8

CHAPTER 2 LITERATURE REVIEW 9

2.1 An ideal collaborative CAD model 9

2.2 The classification of collaborative CAD systems..... 12

2.3 Visualization and annotation 13

2.4 Hierarchical collaborative CAD 15

2.4.1 Multi-representations for features 15

2.4.2 Integration and communication mechanisms for distributed systems 19

2.5 Horizontal collaborative CAD 23

2.5.1 Data exchange	24
2.5.2 Collaborative modeling mechanism.....	29
2.5.3 Run-time architecture... ..	31
2.6 Commercial effort	32
2.6.1 Visualization and annotation tools	33
2.6.2 Co-design tools	33
2.6.3 Groupware tools	35
2.7 Literature on exiting research work	37
2.7.1 Collaborative Solid Modeling	37
2.7.2 WPDSS	40
2.8 Motivation	42
2.9 Issues	42
2.9.1 Feature-level interoperability	43
2.9.2 Data transmission	43
2.9.3 Synchronization and consistency	44
2.9.4 Concurrency	44
2.10 Summary	45
CHAPTER 3 RESEACH METHODOLOGY	46
3.1 Strategy of proposed solution	46
3.2 Mechanism of collaborative design.....	48
3.3 Criteria of client CAD system selection.....	51
3.4 Communication mode.....	52
3.5 Data format for Web -XML.....	54
3.5.1 Features of XML	54
3.5.2 Document object model (DOM)	58

<i>Master Thesis</i>	<i>Table of Contents</i>
3.6 Summary	60
CHAPTER 4 COLLABORATIVE OPERATIONS FOR DATA SHARING/EXCHANGE WITH HETEROGENEOUS CAD SYSTEMS	
4.1 Introduction	61
4.2 Premise — parametric feature modeling	61
4.3 Operation in CAD software	64
4.4 Definition of operation	68
4.5 Requirement of standard operation	72
4.6 Naming mechanism	73
4.7 Conclusions	78
CHAPTER 5 DESIGN OF A COLLABORATIVE SYSTEM	
5.1 Overview	79
5.2 System design	79
5.2.1 System requirements	79
5.2.2 System framework.....	80
5.2.3 Function analysis of the system	80
5.3 Case study	85
5.3.1 Standard operation	86
5.3.2 User interface	89
5.3.3 Demonstration	90
5.3.3.1 Testing part 1	91
5.3.3.2 Testing part 2	104
5.4 Discussion	105
CHAPTER 6 SUMMARY, CONCLUSION & FUTURE WORK	
109	

<i>Master Thesis</i>	<i>Table of Contents</i>
6.1 Summary	109
6.2 Conclusion	110
6.3 Future work	110
REFERENCES	112
Appendix A	123
Appendix B... ..	133
Appendix C.. ..	137

LIST OF FIGURES

Figure 1.1 Traditional product development processes in sequential stages..... 3

Figure 1.2 Product design processes in CE environment..... 4

Figure 1.3 CSCW systems.....5

Figure 1.4 Use of CAD across time & location 5

Figure 2.1 Classification of collaborative CAD 13

Figure 2.2 Multiple feature models of a part across domains 16

Figure 2.3 Unified feature model 18

Figure 2.4 Basic architecture of CORBA 21

Figure 2.5 Proprietary direct translators25

Figure 2.6 Neutral intermediate format translators 25

Figure 2.7 STEP structure27

Figure 2.8 System Structure of MatrixOne36

Figure 2.9 CSM architecture38

Figure 2.10 WPDSS architecture 40

Figure 3.1 Structure of the integrated collaborative design environment 47

Figure 3.2 Schematic information exchange using operations among different CAD systems 48

Figure 3.3 Peer-to-peer vs. client-server topology 53

Figure 3.4 DOM tree 59

Figure 4.1 Classification of form features 63

Figure 4.2 Schematic comparison of traditional and proposed CAD structures..... 65

Figure 4.3 Classification of operations in a CAD system	67
Figure 4.4 Elements of collaborative design	68
Figure 4.5 The class definition for operation objects	69
Figure 4.6 Operation class hierarchy	70
Figure 4.7 Virtual feature mapping table	74
Figure 5.1 System architecture	81
Figure 5.2 Standard operations	87
Figure 5.3 Operations mapping table between Unigraphics and SolidWorks	88
Figure 5.4 The main interface for Unigraphics	90
Figure 5.5 The main interface for SolidWorks	90
Figure 5.6 Demonstration part 1	91
Figure 5.7 The modeling process	92
Figure 5.8 A message about server	93
Figure 5.9 User log on interface within Unigraphics	93
Figure 5.10 User log on interface within SolidWorks	94
Figure 5.11 Screen messages reflected on the server	94
Figure 5.12 The screen layout of the co-design interface at (a) site 0, and (b) site 1	95
Figure 5.13 The screen layout of the co-design interface at (a) site 0, and (b) site 1 and (c) server	98
Figure 5.14 The screen layout of the co-design interface at (a) site 0, and (b) site 1 and server	102
Figure 5.15 The screen layout of the co-design interface at (a) site 0, and (b) site 1	103
Figure 5.16 The part and operations	104
Figure 5.17 The modeling process	105

LIST OF TABLES

Table 2.1 Related work of distributed CE for design and manufacturing 20

Table 2.2 The Communication data format definition 22

Table 2.3 Some mechanisms for collaborative modeling30

Table 2.4 Visualization and annotation tools to assist co-design activities 33

Table 2.5 Co-design tools to share and manipulate real Models 34

LIST OF ABBREVIATIONS

ACIS	application integrated constructs
API	application programming interface
B-rep	boundary representation
CE	concurrent engineering
CAD	computer-aided design
CAE	computer-aided evaluation
CAM	computer-aided manufacturing
CAPP	computer-aided process planning
CAX	computer-aided x application
CIMS	computer integrated manufacturing system
COM	component object model
CORBA	common object request broker architecture
CPU	central processing unit
CSG	constructive solid geometry
CSM	collaborative solid modeling
DDL	data definition language
DML	data manipulation language
DTD	document type declaration
EDM	engineering data management
ERP	enterprise resource planning
EPM	entire product model
HTML	hypertext markup language

IDL	interface definition language
IGES	initial graphics exchange specification
IR	integrated resource
ISO	international standard organization
LAN	local area network
LOD	level of detail
PDES	product data exchange standard
PDM	product data management
PLM	product lifecycle management
RMI	remote method invocation
SDAI	standard data access interface
SET	standard d'Echange et de transfert
SGML	standard generalized markup language
SML	solid modeling library
STEP	standard for the exchange of product model data
TCP/IP	transfer control protocol/internet protocol
UI	user interface
VRML	virtual reality markup language
VDA	Verband der Automobilindustrie
VDAFS	the Transacts VDA surface interface
WAN	wide area network
WPDSS	the web product design support system
WWW	world wide web
XML	extensible markup language

CHAPTER 1 INTRODUCTION

1.1 Background

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 forces companies to develop products 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]. For fast development of complex products in the highly competitive globe market, collaboration among design partners in different locations becomes increasingly important. Over the past decade, there has been a pronounced shift in the focus for designing and implementing support systems from individual decision makers to groups engaged in collaborative work. With the rapid development of IT (Information Technology), CAD (Computer-Aided Design) systems are moving towards supporting distributed and collaborative design. The notion of collaborative design conjures a grand vision of many designers, working at geographically distributed locations, making good design decisions in a reduced timeframe, realizing lower product cost, and producing revolutionary concepts that are never before thought possible [Goldin 1998, Hale 2000].

Effective collaboration and communication are major factors contributing to the timely completion of project. Generally, the technical challenges of collaboration systems include the following aspects:

- *Information sharing*

When collaboration is conducted in a distributed environment, information such as design decisions and ideas is shared among participants of a working team.

- *Communication media*

In the design process, different views and suggestions need to be discussed. Hence designers need to dynamically exchange their opinions either formally or informally using the communication tools integrated into the system.

- *Process management*

Collaborative CAD systems should be able to manage and record information for processes during the product design.

- *Exploration space*

Unlike the traditional product development procedures, e.g. the individual design process, collaborative design environment integrates a wide variety of knowledge and experiences together. Team members are able to contribute ideas, jointly explore design concepts early in the project. Therefore, the exploration space for a design has been expanded. Such a collaborative design environment is useful for the trendy globalization of many enterprises.

However, traditional CAD technologies focused on the interactive process between a computer and a single user. The reason is that the design architectures of the existing CAD systems were based on the computer environments developed in the 1970s. They supported only individual work in resolving the increased complexity in the engineering of new products and production facilities [Kao 1998]. With this constraint, teamed users cannot interact and collaborate effectively and efficiently in editing CAD models. Such CAD systems are stand-alone and installed at a single location for a single user, which restricts users from carrying out group-oriented collaboration. Similarly, the traditional product development is a sequential process that can be described as in Figure 1.1. Because of the lack of collaboration between different applications, product development activities are traditionally serialized. However, such sequential product design, manufacturing planning,

scheduling, and control activities are inefficient for distributed and highly dynamic product development cycles. Stand-alone software systems suffer significantly in today's collaborative global environment. To use such CAD systems for a collaborative project requires common hardware and software since they are entirely platform dependent and single designer centric. This shortcoming makes them incapable of satisfying the flexible requirements of today's collaborative users with their heterogeneous systems.

Figure 1.1 Traditional product development processes in sequential stages

Concurrent engineering (CE) is a systematic approach to integrate design of products and other related processes, including manufacturing and after sale technical support. This approach requires the developers of software solutions to consider all elements of a product life cycle from product conceptualization to disposal, including customer requirements, quality, cost, and schedule [Pham 1998]. CE suggests also tight integration between design of products and manufacturing processes. The conceptual concurrent product development process is illustrated as in Figure 1.2. Concurrent engineering emphasizes the cooperative methodology between the engineering and manufacturing activities which offers real opportunities to create synergy. Successful stories about concurrent engineering usually emphasize the links between many divisions in a company

for sharing and exchanging ideas and experiences. A critical characteristic for the success of concurrent product development is to have team members of different expertise effectively working together on a common task. This is in contrast to a traditional “over-the-fence” or “over-the-wall” working style. With the entire design team organized as a task force, the design engineer and the manufacturing engineer can sit side-by-side and work together. This results in a parallel and continuous development process, rather than a sequential one in which the designs are passed back and forth. Therefore, in companies having multi-location operations, a collaborative CAD/CAM (Computer-Aided Manufacturing) computer system is expected to facilitate a close teamwork between geographically dispersed CAD/CAM engineers. Even within the design process alone, distributed and expanded partnerships among the collaborative players in the market require a design environment with good interoperability to support dynamic project teams.

Figure 1.2 Product design processes in CE environment

Computer Supported Cooperative Work (CSCW) [Greif 1988] is an ideal approach for improving collaboration. CSCW systems are expected to enable the collaboration of users independent of time and location. They are divided into the four categories as shown in Figure 1.3, in terms of time and location that the participants join in the collaboration. Adapted to the CAD domain, these categories can be illustrated as in Figure 1.4.

	Same Time	Different Time
Same location	Face to face interaction	Asynchronous interaction
Different location	Synchronous distributed	Asynchronous distributed

Figure 1.3 CSCW systems

	Same Time	Different Time
Same location	Stand-alone CAD	Shared data + Stand-alone
Different location	Collaborative CAD	Distributed CAD

Figure 1.4 Use of CAD across time & location

Collaborative CAD systems, which are considered as the next generation CAD solutions, should enable collaboration in computer-aided design processes. Unlike conventional CAD systems, a collaborative CAD system allows a group of designers, often geographically dispersed, to work seamlessly with one another over a distributed network during a real-time design process. A remarkable feature of a collaborative CAD system is

that it lends itself to facilitating resolution in the early stages of team design [Chen 2001, Chen 2002] and has gained increasing attention. In the early stage of product development, computer support for collaborative design focused on ways of managing and sharing the design information among a number of professionals involved in a project team. Initial approaches for collaborative design over distance include the use of communication tools such as e-mail, multimedia, shared whiteboard (screen), network audio (conference telephone), and video conferencing. However such groupware tools do not support concurrent interactive cooperative product design. Nowadays the improvement of the interactive collaboration among designers mainly focuses on the development of software tools and methodologies to support collaborative design. Most research concentrates on the real-time collaborative features for general groupware application, such as transmitting, viewing, and annotating 3D models and engineering information. However, based on natural design practice, a collaborative CAD system should not only display CAD models but also support geometric editing. A preferable approach is to enable the modifications of 3D models directly, and display the modifications to all the cooperating designers.

1.2 Objectives

The focus of this project is to develop a collaborative design environment with heterogeneous CAD systems to enable the active participation of various members in a design team either within an organization or between various organizations. The objectives of this research reported herein are as follows:

Objective 1: *To conduct in-depth research on feature-based CAD interoperability in a collaborative engineering environment;*

Objective 2: *To tackle the problem of communication among different CAD systems; and*

Objective 3: *To develop a collaborative 3D CAD system supporting different CAD systems to enable multiple users who design the same part simultaneously and collaboratively at geographically distributed locations.*

1.3 Scope of research project

The scope of the project is as follows:

- To conduct a comprehensive survey on collaborative CAD systems;
- To investigate and develop a system architecture for a collaborative 3D CAD systems supporting different CAD systems;
- To investigate and prototype feature-level CAD interoperability mechanisms for a collaborative application;
- To investigate and prototype concurrency and consistence control mechanisms for a collaborative environment.

1.4 Contributions

The contributions stemmed from this work include:

- 1) Extending the research field of collaborative CAD from a specific CAD system to multiple ones, identifying the modules that an ideal collaborative CAD system should possess, and making the first effort to develop collaborative mechanisms for supporting different CAD systems.
- 2) Development of a novel approach---using *operation* as the basic transmission data unit to address feature-level interoperability and the study of dynamic CAD data sharing/exchange among various CAD systems for collaborative applications.
- 3) Development of the collaborative administrator/controller mechanism supporting the efficiency and stability of a collaborative CAD system.

- 4) The feasible system architecture designed and prototyped to enable collaborative design over the network.

1.5 Thesis Organization

In Chapter 2, the main research issues of collaborative CAD system are surveyed and the existing problems are discussed. Four major issues are identified in developing collaborative design system. In Chapter 3, a command-based solution is proposed to deal with these issues. In Chapter 4, a new method for dynamic CAD data sharing/exchange is proposed and discussed in detail. Then the key mechanisms supporting a collaborative CAD system are proposed and the corresponding detailed functional modules are explained. A prototype system that is developed based on commercial CAD software Unigraphics and SolidWorks is described and discussed with a case study in Chapter 5. Finally, conclusion and future work are discussed in Chapter 6.

CHAPTER 2 LITERATURE REVIEW

2.1 An ideal collaborative CAD model

Collaborative geometric modeling systems are needed to support geographically distributed users such that they can model the same product. They have to work seamlessly with one another over a distributed network during real-time computer-aided design processes; hence, a collaborative CAD system is required. It must possess the features as follows:

- Enabling network communication

Network communication is the key to extend a traditional single-location CAD system to a collaborative CAD system [Kao 1998]. The limitation of a single-location CAD system for multi-location applications is that it does not support the communication channel among the geographically distributed users for necessary communication during the collaboration. Thus a collaborative CAD system must provide an environment for a virtual team. This environment should enable remote designers to work together and to communicate among themselves on a common design activity.

- Enabling real-time high-level CAD data sharing

CAD data is essential contents for engineering communication. To enable remote designers to work together on a common design activity, a collaborative CAD system must provide a virtual environment to share CAD data [Lin 1993, Deitz 1995]. However, the current CAD representations, such as CSG (Constructive Solid Geometry) tree and B-Rep (Boundary Representation), are designed for standalone applications. A 3D CAD file is usually tens or hundreds of megabytes in size, and it poses great difficulty to update a CAD model over a network environment dynamically and efficiently. On the other hand, nowadays, the parametric design and feature modeling technology have been incorporated

into many CAD/CAM (Computer-Aided Manufacture) systems such as Unigraphics, Pro/Engineer, and SolidWorks, etc. Parametric design not only increases the design efficiency, but also makes the updates and modifications of existing designs easier and faster [Roller 1991, Lee 1996]. In design by features, a designer constructs a product model using features, which is considerably faster and easier than working with low-level geometric elements as in geometric modeling [Shah 1995]. Thus, to enable collaboration among remote users effectively and efficiently, a collaborative CAD system must provide high-level CAD data sharing which includes not only low-level geometric model data, but also high-level design information such as features, constraints and parameters.

➤ Providing effective concurrency & synchronization control to support multi-user access and to maintain the consistency of CAD data

When a distributed system supports multiple-user sessions, it is called cooperative application. When the system allows synchronous and concurrent activities, it is referred as a collaborative system.

In the field of collaborative CAD systems, concurrency problems arise when different users try to manipulate one or more objects in the design simultaneously. These concurrency issues are quite complicated, since on a network there is always a delay between the moment that a message is sent and the moment that the same message is received at the other side. Therefore strict locking mechanism needs to be implemented in order to guard the consistency of the product model.

Synchronization involves the handling of changes to the design submitted by the users. These changes will eventually have to be sent to other users to maintain the consistency of product model among remote users. Besides the model, screen views might also need to be synchronized. Synchronization of several distributed applications is also difficult because of the delays on a network. In a synchronized environment, users expect to have the most

recent model available to them. However, the design could have been updated without other users knowing about it. Again adequate locking mechanism can avoid these situations.

➤ Enable feature-level CAD interoperability

The topic of interoperability is closely related to issues of standardization, open systems, cross platform development, and heterogeneity [Mandviwall 1995]. Interoperability of CAD systems in particular, is also an important but complicated problem. The proliferation of CAD systems has led to the need to provide an effective means through which product information can be shared and exchanged among applications and enterprises. This need was recognized two decades ago, and as a result, a number of standards emerged, such as STEP (Standard for the Exchange of Product Model Data), IGES (Initial Graphics Exchange Specification), VDAFS (the Transacts Verband der Automobilindustrie surface interface) and SET (Standard d'Echange et de Transfert), etc [Dong 1997]. However, these standards, which have been used as neutral data interchange formats, are designed for tradition standalone CAD systems. They are limited to support collaborative CAD geometric editing, especially for the real-time 3D CAD. Moreover, these standards are largely used for geometric exchange. Other useful engineering information such as features, constraints and parameters is not well supported. Therefore, feature-level CAD interoperability is still a major research area needed for implementation of the real-time collaborative 3D CAD system. Ideally, such a collaborative design system can provide a high level of communication for product information, including the design intelligence inputted by engineers, i.e. features, constraints and parameters related to 3D models.

2.2 The classification of collaborative CAD systems

Recently, there are active research and developments of methodologies and software tools to support distributed and collaborative design. From the viewpoints of collaborative mechanisms and system architecture, the work can be generally classified into three categories [Li 2004]:

- 1) Visualization and annotation of 3D CAD models to assist collaborative design;
- 2) Establishment of distributed CE environment to support collaborative design with manufacturing analysis and evaluation, and;
- 3) Simultaneous co-modeling of 3D CAD models to implement real collaborative design.

The work in category 1 is mainly to provide a CAD tool for remote users to view and annotate 3D CAD models. The work in category 2 and category 3 also can be called horizontal collaborative CAD and hierarchical collaborative CAD approaches respectively as showed in Figure 2.1. This project mainly focuses on category 3 and provides a fundamental support for category 2.

Figure 2.1 Classification of collaborative CAD [Li 2004]

2.3 Visualization and annotation

The work in the first category is primarily to support visualization, annotation and inspection. To visualize a scene of CAD models to remote users, the 3D parts or assembly models are converted into triangular meshes and organized as a facet-based data structure so as to be displayed and manipulated in a web environment. The main considerations in this category are how to effectively transmit information across networks with the limited bandwidth capability and how to enhance the visualization performance.

There are three ways to improve the efficiency of transmission and to enhance the visualization performance [Li 2004]: (1) using compression techniques; (2) model simplification; and (3) incremental transmission methods.

The compression approach provides a more compact representation of an object. The most common representation used in graphics is mesh. A mesh is represented with topology and geometry properties. The topology is the incidence relation among the vertices. The geometric data describes the location of each individual vertex, and the mesh appearance (normal, color, texture...). Usually, all the information is compressed in a separate way.

Model simplification is another way to relieve heavy data transmission where the level-of-detail (LOD) serves as an option in several 3D streaming tools, such as Hoops [Hoops 2001]. Model simplification can be applied to reduce the geometric details of remote objects that are of little visual importance.

Incremental methods start from a coarse shape and proceed by refinement. Several methods use models like octree representation or subdivision surface, which directly allow such a refinement. Other methods are based on more traditional models like meshes and have to store an initial shape and a set of modifications used for the refinement process. These modifications are computed from simplification techniques applied to the original object [Cignoni 1998].

Since the simplified mesh-based models and the original modeling data structures are either lost or very difficult to recover, the work in the first category can only serve as supporting tools. Simultaneous co-creation and co-modification operations for detailed design cannot be supported effectively.

2.4 Hierarchical collaborative CAD

As illustrated in Figure 2.1, hierarchical collaborative CAD is an asynchronous way to organize a design activity and it should enable communications and interactions among multi-disciplinary engineers. Storing, accessing and maintaining CAD models in consistent repositories securely and conveniently are crucial. The main issues in this category are: (1) multi-representation for features to support multi-disciplinary engineering processes; and (2) integration and communication mechanisms for distributed systems [Li 2004].

2.4.1 Multi-representation for features

In a distributed and concurrent design environment, it is important to provide different views of a part to different application domains [Bronsvoort 2004]. This requires multiple feature models of a part corresponding to different domains. It is because of the domain dependence of features. For example, the solid shown in Figure 2.2 (a) is viewed by a designer as a block with a rib (see Figure 2.2 (b)) but by a process planner as a block with two corner slots (see Figure 2.2 (c)). Here, design and process planning are two domains in which the part is viewed separately. The two different representations are useful in design and manufacturing, respectively. Multi-representation for features allows the geometry/shape of the part to be described at appropriate levels of abstraction as required by a task. The main problems to apply multi-representation are: (1) how to obtain multiple feature representations, and (2) how to automatically propagate feature modifications across different task domains [Jha 2000].

Figure 2.2 Multiple feature models of a part across domains

The problem of identifying multiple feature representations evolved from the need to recognize features that intersect or interact with other features in a part. Many efforts have been reported to extract multiple feature representations. The earliest mention of multiple representations of interacting features was by Vandenbrande and Requicha [Vandenbrande 1993, Chen 1997] and they attribute the existence of multiple feature representations to ‘the non-uniqueness of feature-based representations’. Their papers described a feature recognition algorithm based on hints. The algorithm starts with the `delta_volume` and generates hints based on characteristic patterns in the boundary of the `delta_volume`. As a face in a solid model can be pointing to several features, the `delta_volume` can be decomposed into more than one combination of volumetric features resulting in multiple feature models. The early results in identifying multiple feature representations have been reported by Tseng and Joshi [Tseng 1994]. Their algorithm decomposes the `delta_volume` into cuboidal volumes and then merges these cuboidal volumes in different ways to obtain the multiple machining feature representations. The approaches mentioned are based on the `delta_volume`; they can address only negative features.

Another way to extract required feature model is feature conversion. Feature conversion is to convert features in one domain to the ones in another domain. This approach usually converts the given feature model into an intermediate representation first, and then uses a

feature extraction algorithm to obtain the desired feature model from the intermediate representation. The main problem is that the feature conversion is usually one-way conversion. Thus the modifications on a feature model in one domain can hardly result in changes on the required feature in another domain.

When different tasks use different feature models of the same part, and modifications are made on a feature model as dictated by a task domain, then maintaining the integrity of the part geometry across the different task domains becomes critical. Bronsvoort and Jansen [Bronsvoort 1993] identified the need for two-way conversion among all feature views of a part to support concurrent engineering. De Kraker et al. [Kraker 1995] defined two types of feature conversion and proposed a multi-way conversion system based on constraints maintenance techniques to propagate modifications from one view to other views. They defined a set of validation constraints stored in the form of a graph that consists of disjoint subgraphs. Each subgraph corresponds to one view. The validation constraints were classified into two types—feature validation constraints and inter-feature validation constraints. These constraints have to be defined by the user. Link constraints are then generated between two different views by tracking feature elements in each view that are at least partly on the part boundary and that geometrically overlap. These link constraints are used to propagate modifications made in one view to another view.

My colleague in DRC (Design Research Center), MAE (School of Mechanical and Aerospace Engineering), NTU (Nanyang Technological University), also made contribution in this field. Chen and Ma [Ma 2003, Chen 2004a, Chen 2004b, Chen 2005a, and Chen 2005b] proposed a new unified feature-modeling scheme for information sharing and consistency control among different application feature models, as shown in Figure 2.3. The unified feature model provides a generic feature modeling framework for different application feature models. Although different applications define features in different

ways, their features have 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. With a layer of unified feature model, generic classes as well as the corresponding methods are modeled. Geometry reference is used for connecting higher-level feature specifications with lower-level EPM (Entire Product Model) 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. This scheme is based on feature association and unification concepts, covering three-level geometric and non-geometric relations. Such relations are established and dynamically maintained in the proposed unified feature model. This scheme is intended as the foundation for a future multi-application and feature-object-oriented information infrastructure for collaborative and concurrent engineering.

Figure 2.3 Unified feature model [Chen 2004a]

2.4.2 Integration and communication mechanisms for distributed systems

In a collaborative engineering environment, a framework is needed to integrate remote multi-disciplinary users and various engineering tools such as CAD, CAE (Computer-Aided Evaluation) and CAM. This framework should provide a communication platform to share and exchange information among remote multi-disciplinary users and provide integration mechanisms to integrate different engineering applications. Some significant related work is summarized in Table 2.1.

From Table 2.1, it can be found that two types of communication architecture are employed to provide the communication platform. One is client-server architecture. Client-server architecture offers tremendous opportunities for the sharing of information among product development team members who may be distributed in term of both time and space.

Table 2.1 Related work of distributed CE for design and manufacturing

R & D works	Key characteristics	Communications and infrastructures
Kao [Kao1998]	A collaborative CAD/CAM system extending an existing single-location CAD/CAM system to a multi-location CAD/CAM application.	Exchange data: user defined data as showed in Table 2.2. Client/server under the TCP/IP.
Liu [Liu 2000]	A generic component framework for distributed feature-based design and process planning.	Exchange data: STEP files. Client/server based on MS COM/DCOM.
Chen and Liang [Chen 2000]	A system integrating and sharing engineering information to support CE activities such as domain investigation, functional requirement analysis, and system design and modeling.	Exchange data: STEP AP203. CORBA (Visi Broker)-based web client/server VRML plug-in web browser.
Jacquel and Salmon [Jacquel 2000]	A manufacturing analysis agent system supporting feature-based design and manufacturability evaluation.	Autonomous agent organization (based on Swarm multi-agent system) . ACIS 3D kernel to support design feature agents.
Shen et al. [Shen 2000]	A MetaMorph agent architecture ensuring the coordination among design parts and resource agents to support distributed design and manufacturing activities.	Mediator-centric hybrid agent organization. AutoCAD with AME 2.0 to support product design. TCP/IP protocol to support high-level KQML communication among agents.
Gerhard et al. [Gerhard 2001]	An event-based agent system for distributed design and manufacturing applications.	Agents communicate with each other using event based on Java RMI Agents register themselves with event channels. Event channels provide lookup service and error-handling functionality.
Sung et al. [Sung 2001]	A CyberCut system integrating product design and process planning, including several modules: (1) a Web-based design tool; (2) a new geometric representation for information exchange between the design and process planning modules, and (3) an automated process planning system.	Exchange data: a designed geometric representation called SIF-DSG Web. Client/server.
Sun et al. [Sun 2001], Wu et al. [Wu 2002]	An agent architecture integrating design, manufacturability analysis, process planning and scheduling.	Exchange data: proprietary CAD files Multi-agent organization (based on JATLite multi-agent system). TCP/IP protocol to support high-level KQML communication among agents.
Nidamarthi et al. [Nidamarthi 2001]	Designers upload their CAD files in a server. In a VRML plug-in web, designers can view CAD models and their animation of assembly sequence in VRML converted from CAD files. Designers can download CAD files from the server.	Exchange data: proprietary CAD files, IGES, VRML Web client/server VRML plug-in browser. EAI programming on VRML for the animation of assembly sequence of CAD files.
Tang et al. [Tang 2004]	A multi-application and feature-oriented database framework for web-based CAX application.	Exchange data: STEP. Client/server.

Another architecture is CORBA(Common Object Request Broker Architecture) which is based on agent technology. CORBA is a standard architecture for distributed object systems [Mowbray 1995]. It allows a distributed heterogeneous collection of objects to interoperate. The basic structure of CORBA is illustrated in Figure 2.4. CORBA separates interface from implementation and provides language-neutral types that makes 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 (Object Request Broker). Therefore CORBA can enhance the system integration.

A new information integration platform for computer integrated manufacturing system (CIMS) is presented in [Chan 1999], 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 (Extensible Markup Language)/CORBA based environment for conveying design and manufacture information across traditional technology boundaries.

Figure 2.4 Basic architecture of CORBA

To integrate various engineering tools, the main problem is how to exchange data among different applications. There are two kinds of data exchange, among CAD packages and between CAD application and its downstream applications, such as CAE, CAM.

All the work listed in Table 2.1 used the approach of neutral files as the interchange format to exchange data between CAD application and its downstream applications. However, neutral data interchange formats do not support collaborative geometry editing very well [Kao 1998]. The STEP file format, which had been employed by most research work listed in Table 2.1, cannot support real-time collaborative modeling. Kao [Kao1998] used the user defined data format as shown in Table 2.2 to exchange data. From Table 2.2, it can be found that this data format has certain limitation in its application range, and only tackles curves and surfaces related to design and manufacturing. Some other research efforts had used proprietary CAD files, but these approaches are very limited in their application range because the proprietary data format can only support particular CAD package.

Table 2.2 The communication data format definition [Kao 1998]

Owner ID(IP address)	133.220.19.76	133.220.19.76	133.220.19.76
Command name	INSERT	EDIT	INSERT
Geometry type	NURBI	NURBI	SURCM
Geometry name	12	30	40
Data record stard	40	5	No. of U curves
...	X1 Y1 Z1	X5 Y5 Z5	...

As for the data exchange among CAD packages, very few research work involved this problem. Liu [Liu 2000] proposed a COM (Component Object Model) interface-based framework to wrap and expose API (Application Programming Interface) functions of CAD kernels/systems and process planning modules for remote invocations. The concept of developing standard interface specifications, namely ‘the common core interfaces’, was proposed to encapsulate specific feature functions of different CAD kernels/systems to provide a generic and neutral application layer according to some international standards for features. The advantages of these studies include the straight-forwardness of calling wrapped feature functions and the neutrality of CAD kernels/systems for different applications. However, considering the complexity and variation of features, the programming efforts for implementation are quite huge and the add-on wrapping structures make the system quite heavy. As mentioned previously, the focus of this project is to develop a collaborative CAD system. Data exchange is also the main issue to be tackled. It is expected that this project can provide foundational support for improving the development of the hierarchical collaborative CAD. The issue about data exchange among CAD packages will be discussed in the next section.

2.5 Horizontal collaborative CAD

The horizontal collaborative CAD systems refer to those geometric modeling systems that are capable of supporting various CAD packages, with which participants interact with the system from several geographical locations. Co-modeling functions should be supported by this category of work. This project mainly focuses on the issues in this category of work. In this section, the main issues involved within this category are discussed based on the available literature on existing systems.

2.5.1 Data exchange

The need to exchange or share information between engineering applications has long been recognized [Griffin 1995]; and many researchers keep working on the means for exchanging or sharing the engineering data effectively and efficiently.

Generally speaking, there are two strategies available for those who wish to exchange product data in a computer readable form [Owen 1993, Ma 2000]. They may use [Owen 1993]:

- Proprietary direct translators
- A public domain neutral intermediate (database) format

Figure 2.5 shows the integrated CAD systems using Proprietary direct translators. Figure 2.6 illustrates the structure of public domain neutral intermediate format translators.

In general, when data exchanges take place between n systems, $n(n-1)$ 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.

It can be drawn a conclusion that although direct translator has the advantage of being fast and accurate, it is only suitable for a few of systems and sensitive to the change of version. Whereas, the neutral file format strategy is suitable for the data exchange among a large number of CAD systems. In the direction of the neutral file format strategy, a number of standards emerged, such as STEP, IGES, VDAFS and SET.

Figure 2.5 Proprietary direct translators [Owen 1993]

Figure 2.6 Neutral intermediate format translators [Owen 1993]

➤ 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, the experience of transferring

CAD data via IGES format in the past years has clearly identified its deficiencies and weaknesses. First of all, the sizes of IGES files and consequently the processing time have been recognized as a significant application problem. Secondly, the double pointing mechanism creates inconsistent errors in the implementation of processing programs. The absence of clearly defined subsets and guidelines causes compatibility problems when using IGES. Thirdly, in the IGES specifications, some geometric definitions for curves and surfaces are based on polynomial representations. Small numerical errors in the coefficient values sometime result in large errors in the geometrical shape.

➤ 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 (the Transacts Verband der Automobilindustrie Surface Interface) was developed primarily for German car manufacturers as a neutral description of free form surfaces [Papavasiliou 2002] 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 results in fewer 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 is lost.

SET (Standard d'Echange et de Transfert) 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].

➤ STEP

ISO (International Standard Organization) 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 [ISO 1994a].

Basically, STEP covers geometry, topology, tolerances, relationships, attributes, assemblies, configuration, etc. STEP has been divided into multiple parts to achieve this ambitious goal. Figure 2.7 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 2.7 STEP structure [ISO 1994a]

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 [ISO 1994b], which is used to specify the representation of product information.

As for collaborative CAD system, to support different CAD packages, a mechanism is needed to enable the communication among different CAD packages. As discussed in Section 2.1, this mechanism should support real-time high level CAD data sharing. However, the existing data exchange strategies and the standards mentioned above can not meet the requirements of collaborative CAD system because of the following reasons:

- Cannot support real-time collaborative modeling.

The data exchange strategies and standards mainly focus on CAD files format translation for traditional stand-alone CAD systems. At present, these strategies and standards can only establish a connection among various CAD files, but not the connection among CAD running sessions. Applying these strategies and standards during a collaborative modeling, one designer should translate the native CAD file into a neutral file for sharing CAD data with designer using different CAD packages. If some modifications are made, the whole neutral file needs to be regenerated. These processes are asynchronous. Thus the data exchange strategies and standards can only set up an indirect and asynchronous channel among various CAD packages and cannot support real-time collaborative modeling.

- CAD data is stored in a file format.

The CAD data sharing is a file level sharing. Obviously, this file level sharing mechanism cannot support real-time applications. Moreover, this file level sharing leads to difficulties in maintaining the consistency of CAD data because each designer can only manipulate the CAD data in the native file.

- Information loss.

Nowadays, feature modeling and parametric design methods become more and more important to express design intent and to accelerate design process. However, the current data exchange strategies and standards are developed for geometric data exchange. In the process of data exchange, useful information such as features and parametric information is often lost. Hence, such data exchange is not complete.

In conclusion, for supporting collaborative modeling, a new mechanism should be established to enable real-time high-level CAD data sharing among various CAD packages.

2.5.2 Collaborative modeling mechanism

To enable collaborative modeling among remote users, three common collaborative modeling mechanisms have been developed, namely modeling client/communication server, manipulation client/modeling server and application or service sharing. In terms of information technology, they also can be called strong client/lean server, lean client/strong server and point-to-point respectively. Their characteristics are summarized in Table 2.3.

Table 2.3 Some mechanisms for collaborative modeling

Mechanisms	Functional descriptions	Illustrative diagrams	Characteristics	Related research Work
Modeling client+ Communication server	<p>Clients are equipped with whole CAD systems and some communication facilitators.</p> <p>A server plays as an information agent and exchanger to broadcast CAD files and commands generated by a client to other clients.</p>		<p>Standalone CAD systems can be conveniently distributed through this mechanism. Due to the heavy-weighted client mechanism, it is hard to be migrated to web applications.</p>	<p>CollabCAD[Collab 2001], Nam and Wright [NAM 1998], Qiang et al. [Qiang 2001], Pahng et al. [Pahng 1998].</p>
Manipulation client+ Modeling server	<p>The data structures in clients are light-weighted and they primarily support visualization and manipulation functions.</p> <p>The main modeling activities are carried out in a common workspace in the server side.</p>		<p>Modules can be rent out as an application service provider (ASP). Data consistency is easily kept since the primary models are created and maintained in the server.</p>	<p>AlibreDesign [Alibre 2001], GS-Design [GS-Design 2001], Li et al. [Li 2002], OneSpace [OneSpace 2000], van den Berg E et al. [Van 2000], Tay et al. [Tay 2003], Rafael et al. [Rafael 2001], Ouyang et al. [Ouyang 2004].</p>
Application or service sharing	<p>The services or modules of a system can be shared and manipulated by other systems.</p> <p>For the Inventor collaborative tool, an MS Net meeting tool is embedded for application sharing.</p>		<p>This mechanism enables a convenient manipulation on remote services or applications. Due to the heavy burden of networks, the manipulation efficiency of systems is low.</p>	<p>Begole et al. [Begole 1997], Inventor collaborative tool [Inventor 2001].</p>

2.5.3 Run-time architecture

For real-time distributed multi-point applications, where two or more people in different locations are able to share and manipulate their on-line work, run-time architecture is needed to manage the run-time system. Generally, there are two kinds of typical run-time architecture: centralized architecture and replicated architecture [Greenberg 1994, Dourish 1996, Roseman 1996].

Centralized architectures use a single application program, residing on one central server machine, to control all input and output to the distributed participants. The advantage of a centralized scheme is that synchronization is easy—state information is consistent since it is all located in one place, and events from the client processes are always handled in the same order because they are serialized by the server. The main problems are latency and bottlenecks. First, a centralized scheme implies sequential processing, where user's input is transmitted from the remote machine to the central application, which must handle it and update it before the next input request can be dealt with. If the system latency is low, this is not a problem. But if it is high, the entire system will become sluggish. While sluggishness is annoying when others' actions are delayed, it is devastating when the system is unresponsive to a person's own local actions, especially in highly interactive applications. Second, the central system can become a performance bottleneck. Highly interactive and graphical applications can push even the fastest CPUs (Central Processing Unit) to their limits when several screens must be updated. Similarly, relaying of all activities to and from a single process can create a traffic jam in some environments.

Replicated architectures, on the other hand, execute a copy of the program at every site. Thus each replica must coordinate explicitly both local and remote actions, and must attend to synchronizing all copies so they do not get out of step. A replicated scheme implies

parallel processing, where the handling of interactions and model updates can occur in parallel at each replication. If done properly, communication is efficient, as replicas need only exchange critical state information to keep their models up to date. While remote activities may still be delayed, a person's local activities can be processed immediately. Process bottlenecks are less likely—each replica is responsible for drawing only the local view, unlike the central model, which must update the model of all participants. The main problem of replicated architecture is the complexity for synchronization and concurrency control.

So far, there is no real answer to whether a centralized or replicated scheme works best for distributed system. Replicated systems are probably better for some projects with larger groups, or slower networks, or for applications that demand local responsiveness. A centralized system would likely work just fine for a very small group of users, given a high bandwidth, low latency network and an application that makes only modest demands to the processor. No matter which architecture is employed, to develop a useful collaborative CAD system, both synchronization and concurrency problems have to be solved. Synchronization involves the handling of changes to the design caused by one of the users. These changes will eventually have to be sent to other users to maintain the consistency of the whole product model. In the field of collaborative CAD systems, concurrency problems occur when different users try to manipulate one or more objects in the design simultaneously.

2.6 Commercial Effort

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

2.6.1 Visualization and annotation tools

The characteristics of some commercial systems are summarized in Table 2.4. These systems can support an on-line team that is concerned with a macro-view of a product, customer survey for new products and conceptual design. Simultaneous co-creation and co-modification operations cannot be supported effectively.

Table 2.4 Visualization and annotation tools to assist co-design activities

Products	Characteristics and functions	Information distributed methods	Support file formats
Cimmetry systems Autovue [Cimmetry Systems Autovue 2006]	(1) A viewer for part and assembly models, (2) View, make-up, measure, explode, cross-section, etc.	3D streaming	STEP, IGES, STL, SAT, XMT, VRML, DXF/DWG.
InFlow ConceptWorks [ConceptWorks 2006]	(1) An add-in viewer to SolidWorks, (2) View and mark-up.	3D streaming	SolidWorks compatible files.
Actify SpinFire [Actify 2006]	(1) A viewer for part models, (2) View for native or simplified SolidWorks files.	Download	STEP, IGES, STL, SAT, XMT, VRML,
Autodesk Streamline [Streamline 2006]	(1) A platform based on the VizStream, (2) View, measure, bill of materials.	3D streaming	Inventor compatible files
SolidWorks eDrawing [eDrawing 2006]	(1) A viewer for native of simplified SolidWorks files, (2) View, mark-up, measure, 3D pointer, animation.	Download	SolidWorks compatible files.
Hoops Streaming Toolkit [Hoops 2001]	(1) A toolkit to provide 3D streaming APIs, (2) BaseStream class library, advanced compression, attribute (color, texture) support object prioritization, etc.	3D streaming	SAT, XMT, HSF.
Adaptive Media Envision3D [Adaptive 2006]	(1) A viewer for part models, (2) View, mark-up, redline, chat.	3D streaming	Stereo lithography, VRML, DXF/DWG.

2.6.2 Co-design tools

The characteristics of some commercial systems are summarized in Table 2.5. Real co-modeling functions can be supported by these systems.

Typically, these systems in Table 2.5 consist of four kinds of components: 1) team management, 2) part and assembly modeling, 3) repository, and 4) messaging. These

components can be used to establish a distributed workspace with effective sharing of detailed design models in a well-organized team.

Table 2.5 Co-design tools to share and manipulate real Models

Products	Collaborative mechanisms	Functions	Supported file formats
Alibre Design [Alibre 2001]	<i>Team design sessions:</i> a design session can be used to organize a virtual team to design 2D and 3D models simultaneously. <i>Repository:</i> through repository, user's models can be securely shared and accessed. <i>Message centre:</i> it can support message sharing among users.	Modeling, mark-up, annotation, view, text and voice chat, directly transferring CAD models, peer-to-peer communication.	STEP, IGES, STL, SAT.
Cocreate OneSpace [OneSpace 2000]	<i>3D personal collaboration:</i> through this service, up to two other users can be invited to an online meeting. Meeting users can view and mark-up 2D and 3D models. <i>Model manager:</i> it can store and share user's models through a database, and specify who has permission to read and modify design work. <i>Project data manager:</i> it can organize 2D or 3D project files in a database and helps track of document version and history.	Modeling, mark-up, view, net meeting, integration with PDM, directly transferring CAD models, client/server commutation.	STEP, IGES, IDF, EMatrix, Word, PDF, zip.
Collab-CAD [Collab 2001]	<i>Design team:</i> members in the team can simultaneously design and share 2D and 3D models. <i>Repository:</i> it can store and share user's models through a database.	Modeling, text or voice chat, directly transferring CAD models, client/server communication.	STEP, IGES, VRML.
Autodesk Inventor Collaborative Tool [Inventor 2001]	<i>Application sharing:</i> MS net meeting tool is embedded into Inventor systems to organize co-design activities. An Inventor that has the 'control baton' can control and manipulate another remote Inventor system to design and the controlled system is an observer. The control baton can be acquired and exchanged.	Modeling, net meeting, whiteboard, chat, directly transferring CAD models.	Inventor compatible files.

However, owing to commercial restrictions, it is difficult to obtain the technical details and internal mechanisms of these systems. The above co-modeling systems can only support one specified CAD system during the collaboration design process. After collaboration, like traditional CAD system, an interface based on some neutral file formats is provided to exchange information with other CAD systems. Another main problem for the above co-modeling systems is that communication efficiencies are still quite far from satisfactory when large-size parts or assembly models are designed collaboratively.

2.6.3 Groupware tools

Windchill is the PDM (Product Data Management) product provided by PTC (Parametric Technological Company) [Windchill 2006]. Windchill supports distributed product development and enables real-time collaboration within enterprise or across enterprise. Manufacturing engineers can see the released versions while the design engineer is looking at the engineering change orders 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.

ENOVIA MatrixOne is a recognized leader in delivering collaborative PLM (Product Lifecycle Management) solutions [MatrixOne 2006]. MatrixOne delivers the industry the most flexible PLM environment that has been built with the needs and challenges of the global enterprise in mind. The functionality included in MatrixOne spans the entire product

development lifecycle, providing companies with the ability to rapidly adjust business process functionality to meet their unique needs today and tomorrow, at the lowest possible total cost of ownership. MatrixOne provides companies with a single, secure environment that eliminates the barriers caused by geographically dispersed organizations and value chains, multiple disparate systems and increasing security requirements. As a result, MatrixOne enables teams to work, collaborate and share ideas effectively regardless of location, all without jeopardizing the security of their company’s intellectual property.

The MatrixOne Environment consists of three very powerful components that work together to enable organizations to harness the power of their value chains as shown in Figure 2.8.

Figure 2.8 System Structure of MatrixOne [MaxtrixOne 2006]

The collaborative design functions are mainly support by the MatrixOne design central. The MatrixOne design central enables designers around the world to manage and

collaborate from one application, no matter what design tools they are using. Designer Central eliminates the need for multiple CAD work group managers and the cost associated with supporting each related data manager. The main functions of the design center are as follows [Matrix Design Central 2006]:

1. Provide work-in-process data management across all popular mechanical and electrical design tools; and
2. Visually collaborate around the design information or other information related to the design; and
3. Schedule meetings for review at any step in the development process; and
4. Create private and public workspaces for organizing and sharing designs; and
5. Securely provide design information to suppliers and customers.

From the above introduction, it can be found that MaxtrixOne mainly focus on data management. It is not clear whether MaxtrixOne can support co-modeling functions.

2.7 Literature on exiting research work

In this section, some literature on existing collaborative systems is studied and some general issues involved in development are discussed.

2.7.1 Collaborative Solid Modeling [Chan 1999]

The Collaborative Solid Modeling system (CSM) reported by [Chan 1999] allows multiple remote users to design a complex solid model collaboratively and concurrently via the Web.

- **Development environment**

The model kernel is constructed on the Solid Modeling Library (SML) that is a Java call library [Mantyla 1995]. And Java RMI (Remote Method Invocation) is employed to enable network communication.

- **Run-time architecture**

The outline of CSM structure is shown in Figure 2.9. The replicated architecture is adopted with this system. A shared model can be accessed and modified by multiple users, while every user actually has his/her own local model.

Figure 2.9 CSM architecture [Chan 1999]

- **Collaborative modeling mechanism**

In CSM, the collaboration mechanism with modeling client and communication server is applied. When several distributed clients want to start a modeling session, Java-applets that run on those clients connect to the CSM server, while this server is responsible for synchronizing the clients.

- **Synchronization**

Changes invoked by any client are first applied to the local copy of the model. After that, the changes are sent to the server, which updates the global model at the server-side. Finally the changes are sent to all other clients to update their local copies.

- **Concurrency control**

CSM uses two methods for concurrency control: TOKEN or FUNCTION. When TOKEN is set, the system uses simple token passing and only the user with TOKEN can edit the shared model. FUNCTION means particular functions can be blocked for some users. Examples of such functions are *Add* and *Delete*.

- **CAD data exchange**

The basic representation within the system is Constructive Solid Geometry (CSG). For viewing, the CSG-model is converted into a Boundary Representation. The CSG-representation is used for sending changes incrementally over the Internet. CSM is capable of importing and exporting STEP intermediate files, so it can be used in combination with other major design and manufacturing applications.

- **Shortcomings**

Although the CSG-representation is used for sending changes incrementally over the Internet instead of Boundary Representation, the network-load is still very high. Moreover, some useful information such as parametric information will be lost. As for concurrency control, the FUNCTION approach still allows local inconsistencies. For example, when two clients change the same object at the same time, assuming they both have the necessary rights, the local models would be different for those two clients. A better solution would be a locking mechanism, but that has not been implemented yet for this system. Another problem is that CSM cannot support different CAD systems during a real-time collaborative modeling process. Although, CSM can import or export STEP file, as discussed in Section 2.4.1, the importing or exporting STEP file is an asynchronous process and cannot meet the real-time requirement.

2.7.2 WPDSS [Qiang 2001]

The Web Product Design Support System (WPDSS) [Qiang 2001] extends the traditional stand-alone CAD software to a multi-location application for collaborative product design through the Internet. The architecture of WPDSS is illustrated in Figure 2.10.

Figure 2.10 WPDSS architecture [Qiang 2001]

- **Development environment**

The system was coded in Java. The model kernel is commerce CAD software- Unigraphics and TCP/IP (Transfer Control Protocol/Internet Protocol) protocol is employed to support network communication.

- **Run-time architecture**

Again, the replicated architecture is used with this system. There is no shared model. Each user maintains the local part file and after completing the design, the part file is uploaded to the server and stored in the server file system.

- **Collaborative modeling mechanism**

Modeling client/communication server mechanism is applied in WPDSS. When the users start collaboration, they need to download the co-modifying interface program from the WPDSS server, and run it on their local workstations. The co-modifying monitor on the server administrates the group co-modifying processes.

- **CAD data exchange**

To reduce network load, The WPDSS uses the macro functions in data transmission, because they are useful for performing short, repetitive tasks, and can be tailored to the working environment. The obvious advantage of writing a macro to perform a task that would otherwise be done manually, is automating the mechanical and repetitive tasks. The size of a macro file is much smaller than a CAD geometry file. Macro files reduce the data to be transmitted substantially since they only record the interactions with the software interface. The operation procedures can be repeated on the same CAD software. Since it does not transfer complex geometry data, the data information will not be lost during transmission.

- **Synchronization**

Using macro functions as the transmission media has increased the complexity of synchronization because the macro files must be executed from the same point in the interface from which they are created. It cannot be executed if the CAD operation sequence is random. In WPDSS, before a client to modify the model, he needs to send a request to the server. The request is assigned with a logical_clock according to the time that has been transmitted. The server always “listens” to incoming requests, and checks their logical_clock. When the server receives more than one request, according to the principle of “first come, first served”, a request is chosen to execute. Then the server gives the corresponding client a reply and other requests from other clients are rejected. The

client that received the reply will modify the model and use the macro file to record the modifications, and then transmit the macro file to the server. The server propagates this macro file to other clients. The other clients receive the macro file and execute the macro file to update their local models. By employing this function, a nearly real-time co-modification process can be realized.

- **Shortcomings**

Using macro commands as the transmission data, the user can avoid manipulating low-level geometric data and the communication efficiency can be improved greatly. On the other hand, the WPDSS is developed based on Unigraphics and the powerful modeling function of Unigraphics is inherited. However WPDSS cannot support other CAD packages because the macro operations can only be replayed on workstations using the same CAD software.

2.8 Motivation

In practice, several engineers are involved in the development of a product. More often, product development teams nowadays involve multiple parties, who are often geographically distributed. Also, collaboration among team members plays an increasingly important role in solving design conflicts as early as possible in different design stages. Thus, an environment that supports collaborative design with current popular CAD systems is a desideratum.

2.9 Issues

Four major issues are identified in developing the Internet based collaborative design system. They are: 1) feature-level interoperability; 2) data transmission; 3) synchronization and consistency; and 4) concurrency.

Except the issue of feature-level interoperability, other issues may also be encountered in those collaborative systems based on one specific CAD system. However, in this project, these issues are more complicated because they should be addressed in an environment that supports different CAD systems.

2.9.1 Feature-level interoperability

In a collaborative product development environment, individual companies, or teams involved in a given product development activity should be able to use a set of heterogeneous systems, and still exchange information meaningfully and pass knowledge between various phases. Hence the research scope is extended from one specific CAD system to multiple ones. This is hard to realize because of the lack of interoperability among various systems.

Thus to develop the targeted system, interoperability has to be addressed.

2.9.2 Data transmission

Collaborative design is a highly interactive process and the performance over the network is a key factor to evaluate a collaborative CAD system. Owing to large size of CAD files, file transmission through Internet could be very time-consuming and unreliable. It is intolerable for a designer to wait for a long time for updating large-size CAD models across networks.

To reduce network-load, an appropriate way to represent CAD data is needed. Ideally, collaboration communication should not only contain CAD information and convey design intent, but also have the size of data packets as small as possible.

Incremental transfer is an alternative way to reduce the network-load. The design process is iterative. If a designer makes some modifications, just the modifications are

transferred incrementally, instead of the whole CAD file. The network-load could be greatly reduced.

2.9.3 Synchronization and consistency

Synchronization involves the handling of design modifications by any user. These changes will eventually have to be sent to other users to maintain the consistency of CAD models.

In this project, it has been assumed that there are different CAD systems employed to carry out modeling functions in collaboration. The formats of geometric models created by different CAD systems are different. Thus two aspects need to be considered in order to address the issue of synchronization and consistency. Firstly, differently formatted geometric models are incomparable, and then the problem is how to determine whether these models are synchronized and consistent. Secondly, when some modifications are made on one geometric model, how the corresponding modifications could be made on other differently geometric models in order to maintain consistency.

2.9.4 Concurrency

Concurrency problems arise when different users try to manipulate one or more objects in the design simultaneously. If this issue is not addressed properly, serious problems can arise, such as inconsistency of geometric model, or processes indefinitely waiting for each other, i.e. deadlock.

An effective mechanism is needed to detect and solve the concurrency conflicts among various operations performed on the model from various parties during the design process.

2.10 Summary

This chapter first discussed the requirements of an ideal collaborative CAD system. Based on the related research studies, the classification of collaborative CAD is given. It can be generally classified into three categories. The first category focuses on visualization and annotation. The functions and the ways to improve the efficiency of transmission and to enhance the visualization performance are discussed. For the second category-hierarchical collaborative CAD, the two main issues, (1) the multi-representation for features to support multi-disciplinary design and (2) the integration and communication mechanisms for distributed systems are discussed. For the third category - horizontal collaborative CAD, the issues involved with developing a collaborative co-modeling system are investigated. The literature on existing systems is studied. At last, based on the literature review, the key issues to develop a collaborative system supporting multiple CAD systems are identified. They are feature-level interoperability, data transfer, synchronization and consistency, and concurrency.

As reviewed in this chapter, it is found that few systems can support various CAD systems during collaborative design. Usually, only an interface is provided for format exchange between CAD packages based on some standards. The synchronization and concurrency control are not provided or still need to be improved. Therefore, this project focused on developing a collaborative CAD technology supporting different CAD systems to enable multiple users to design the same part simultaneously and collaboratively at geographically distributed locations.

CHAPTER 3 RESEACH METHODOLOGY

A command-based integration solution is proposed to deal with the issues identified in Chapter 2. The structure of the proposed system is shown in Figure 3.1. For every client CAD system, a manager application is to be developed to encode and decode the collaborative operations of a product model, where the commands are encoded by XML format to be transferred. Compared with other collaborative design solutions, e.g. CMS [Chan 1999], webspIFF [Rafael 2001], the implementation of the proposed method involves only integration work; therefore, the platform can be constructed without going deep into the kernel of the selected CAD systems, and the powerful functions of the selected systems greatly benefit client users.

3.1 Strategy of proposed solution

The main idea of the proposed solution is to utilize the available resources of different CAD systems as much as possible. As shown in Figure 3.1, manager applications are developed on top of the selected commercial CAD systems; they carry out the functions of data encoding, data transferring, and data decoding, which are required for collaboration between users and systems.

Figure 3.1 Structure of the integrated collaborative design environment

The proposed solution uses *operation* as the basic transmission data unit to realize CAD data sharing/exchange among different CAD systems. The details on operation definition, method, and implementation are described in Chapter 4. As illustrated in Figure 3.2, during the process of collaborative design, once a designer with CAD system A makes some modifications, the corresponding operations, namely local operations A, are captured and mapped into standard operations by the manager A. Then such standard operations are encoded into XML format and transferred to the server as a packet. On the site of CAD system B, when the command packet is received, the manager B first decodes it into standard operations, and then maps them into the local operations which are executable

operations for CAD system B. Through executing local operations by CAD system B the same modifications can be achieved on CAD file B. In this way, the CAD file B is updated.

Figure 3.2 Schematic information exchange using operations among different CAD systems

By this method, no special collaborative CAD system is to be developed; rather, only attached manager applications are to be developed on some selected CAD systems.

3.2 Mechanism of collaborative design

Generally, there are two possible approaches to resolve design operation conflicts in a collaborative design process. The first approach is the master-slave style in which one master user takes the responsibility of controlling the entire system to interact with the application and the other users may only use the result. This approach is clearly limited since users are discouraged from participating in the collaborative design process. Another moderated approach is based on a token-based model that allows users to request and pass the manipulation control to each other, which is similar to passing a pen in a meeting. Technically, a token, which is a special bit pattern, travels around the circle. To send a

message, for example, a computer catches the token, attaches a message to it, and then lets it continue to travel around the network.

After a detailed analysis of the existing control mechanisms and making a feasibility study for the CAD system, a mechanism called administrator/controller is proposed, which is actually a combination of both the master-slave and token-based mechanism.

As showed in Figure 3.1, assuming that there are n users involved in the collaborative design, among them, there is one administrator and one controller. The administrator is the user who initializes the collaboration space, and the controller is the user who can modify the product data. The administrator and the controller can be the same user or different users. During the collaboration design, there is only one administrator while the controller can be assigned to different users by permission of the administrator. When the administrator creates a collaboration space, the manager application on his machine delivers the product data to all the other $n-1$ users through the manager server. In other words, the initial product data is transferred from his machine to the server first; and then the server transfers the data to the machines of other $n-1$ users.

By default, the administrator and the controller are the same person, but any one who would like to modify the model can ask the administrator to assign him the control right. If the control right is assigned, he becomes the controller. To let the administrator have better control during the collaboration, the administrator can withdraw the control right at any time. Any operation on the machine of the controller will be captured by the manager application on his machine and be encoded into a command packet in XML format; the packet is first sent to the system server; then, the server delivers the command packet to all other machines. On each of the other machines once the command packet is received, the manager module decodes the packet and applies the operations on the product model on its own machine. When the update is finished, every local machine sends a confirmation to

the server. If the server does not receive the confirmation from any machine, the command packet will be sent to the machine two more times. If there is still no confirmation from the machine, the user is assumed to have left the collaboration space. The user on that machine needs to re-join the collaboration space. Every time a new user joins the collaboration space, the current product data on the machine of the administrator is utilized to initialize the product model for the new user.

To achieve concurrency and consistency, there are four potential levels for collaboration: assembly level, part level, feature level and geometrical entity level. Assembly level collaboration has been largely achieved by the existing PDM (Product Data Management) systems such as iMan, TeamCentre, etc. In the part level, to perform modeling operations, the user has to lock the model. As the whole part is locked and excluded from other users, the part level collaboration design will have a very strict concurrency handling policy to prevent design conflicts. However, such a locking method may limit the design flexibility. On the contrary, working at the geometrical entity level may have the most freedom for design, where the user needs not lock the whole part before operation; however the server should check any possible conflict caused by collaborative users. Such level concurrency can only be loosely controlled and will possibly affect the collaborative design performance because the checking system implementation will be very complex.

Comparing the above two levels, a good compromise is the feature level concurrency policy. The feature level for collaboration is higher than the geometrical level and lower than the part level. In the feature level, the locking granularity is with features, and a user needs to lock only involved features. These features can still be viewed by all users but other users cannot modify them until the original user unlocks them. Such a scheme can prevent design conflicts more effectively. Moreover, compared to the part level

concurrency control, the feature level would not limit the design concurrency too much. Ideally, a flexible feature level concurrency control should be employed. However, feature interference checking methods are required among participating CAD systems. In this work, the author focuses on the part-level collaboration prototyping, and leaves other level mechanisms for future research.

3.3 Criteria of client CAD system selection

The integration solution presented here attaches a manager application on the client CAD system, thus there are two major requirements for each client CAD system:

1. The system has the ability to be programmed; and
2. Every operation performed on the user's machine can be traced.

For the programmable requirement, most current popular CAD systems have the user programming interfaces (e.g., SolidWorks and Unigraphics). There are two ways to make the CAD systems run automatically– by script language or by API (Application Programming Interface). Script languages are often interpreted languages, so they run a lot slower than an equivalent API program like in a C++ program. For example, a C++ program is compiled from the source code to generate actual native machine instructions, and it runs faster. Script languages cannot provide the communication functions, but a C++ program can have the communication ability by using some existing libraries. Therefore, in the integration solution, the API mode is chosen.

The second requirement is that the API program must be able to trace every operation performed on the system by users. According to the mechanics of the solution introduced in the above section, every operation on the system of the controller should be encoded. Thus, the selected CAD systems must provide the ability for an API program to trace all the operations of the system in real-time. Every operation should also have the

corresponding API functions, so that users on all sites can apply their API functions to update their local models. If not all the parties have the required API functions, then only those with the required API functions are allowed in the collaborative design environment.

3.4 Communication mode

From the mechanisms of the proposed solution, clearly, the product data is stored on every client's machine. The central server does not maintain the full set of the product data – only the updated data is delivered through it from the controller to the other clients. In this structure, it plays like a valve instead of a storeroom.

When conducting the communication to transfer data, there are two basic ways to structure the messaging topology of a communication procedure: peer-to-peer and client/server (as illustrated in Figure 3.3). The implementation of the peer-to-peer is simple; however, the client/server mode is more efficient than the peer-to-peer mode, especially for the case with a lot of clients involved in the design. In particular, it scales much better than peer-to-peer mode because additional users cause only a linear increase in the messaging traffic. The processing power of a user's computer will greatly influence the responding speed in the peer-to-peer mode, but have almost no effect in the client/server mode. In the integration solution, the client/server topology is selected.

Figure 3.3 Peer-to-peer vs. client-server topology

An application of XML is defined via a document type definition (DTD). A DTD provides a list of elements, attributes, notations, and entities contained in a document, as well as their relationships to one another. DTD specifies a set of rules for the structure of a

document. For example, it defines exactly what is and is not allowed to appear inside a document. These characteristics make XML become a perfect candidate for textual representation of the commands. Both the initial standard operations and the later update standard operations transferred between the clients and the server are represented by the XML scheme. The standard operation is encoded into XML by the manager application attached on every client's machine.

3.5 Data format for Web –XML [Harold 1999]

XML is a simple, very flexible text format derived from SGML (Standard Generalized Markup Language) [ISO 8879], which is designed to enable interchange structured documents over the Internet.

3.5.1 Features of XML

XML is in fact a simplified subset of SGML that maintains the SGML features for extensibility, structure, and validation [Bray 1997]. It has the advantages of:

- Extensibility—users can define new elements, containers and attributes at will;
- Structured Data—since a DTD can constrain the information model of a document;
- Validation—every document can be validated.

➤ XML Markup

XML documents are composed of markup and content. There are six kinds of markup that can occur in an XML document: elements, entity references, comments, processing instructions, marked sections, and document type declarations. The following sections introduce each of these markup concepts.

- **Elements**

Elements are the most common form of markup. Delimited by angle brackets, most elements identify the nature of the content they surround. Elements may be empty, as mentioned above, in which case they have no content. If an element is not empty, it begins with a start-tag, `<element>`, and ends with an end-tag, `</element>`.

- **Attributes**

Attributes are name-value pairs that occur inside tags after the element name. For example, `<div class="preface">` is the div element with the attribute class having the value preface. In XML, all attribute values must be quoted.

- **Entity References**

In XML, entities are used to represent special characters reserved to identify the start of markup. Entities are also used to refer to often repeated or varying text and to include the content of external files. The name of the entity must be unique since it is used for reference.

- **Comments**

Comments begin with `<!--` and end with `-->`. Comments can contain any data except the literal string `--`. Comments are not part of the textual content of an XML document. An XML processor is not required to pass them along to an application.

- **Processing Instructions**

Processing instructions (PIs) are also not textual part of the XML document, but the XML processor is required to pass PIs to an application to provide processing information.

- **CDATA Sections**

In a document, a CDATA section instructs the parser to ignore most markup characters.

➤ **Document Type Declarations (DTD)**

XML is extensible since it allows users to create new tag names with declarations. Declarations allow a document to communicate meta-information to the parser about its content. There are four kinds of declarations in XML: element declarations, attribute list declarations, entity declarations, and notation declarations.

- **Element Declarations**

Element declarations identify the names of elements and the nature of their content. There are four types of content models: EMPTY indicates that the element has no content, ELEMENT ONLY indicates that the element contains only element, MIXED indicates that the element contains both element and character data and ANY indicates that any content is allowed. Here is a complete set of element declarations:

```
<!ELEMENT movie (title, director+, actor*, comments?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT comments (#PCDATA)>
```

- **Attribute Declarations**

Attribute declarations identify which elements may have attributes, what attributes they may have, what values the attributes may hold, and what default value each attribute has. A typical attribute declaration looks like this:

```
<!ATTLIST oldjoke
  name ID          #required
  label CDATA      #implied
  status ( funny | notfunny ) 'funny'>
```

- **Entity Declarations**

Entity declarations allow you to associate a name with some other fragment of the document. That construct can be a regular text, a document type declaration, or a reference

to an external file containing either text or binary data. There are three kinds of entities: internal entities, external entities and parameter entities.

Example of internal entities:

```
<!ENTITY ATI "ArborText, Inc.">
```

Example of external entities:

```
<!ENTITY boilerplate SYSTEM "/standard/legalnotice.xml">
```

Example of parameter entities:

```
<!ENTITY ATIllogo SYSTEM "/standard/logo.gif" NDATA GIF87A>
```

- **Notation Declarations**

Notation declarations identify specific types of external binary data. This information is passed to the processing application. A typical notation declaration is:

```
<!NOTATION GIF87A SYSTEM "GIF">
```

- **Validity of XML document**

In order to check if an XML document is valid, the XML processor first reads the whole DTDS, internal DTD and external DTD in sequence. Then, it checks the XML document whether it obeys the syntax of XML. A well-formed document is valid only if it contains a proper document type declaration and if the document obeys the constraints of that declaration (element sequence and nesting is valid, required attributes are provided, attribute values are of the correct type, etc.). The XML specification identifies all of the criteria in detail.

- **XML Linking**

The XML linking specification (XLL) introduces a standard linking model for XML. A link expresses a relationship between resources. XLL specifies how to control the link over the semantics. It supports not only simple links like in HTML (Hypertext Markup Language), but also extended links which can involve more than two resources. In addition,

XLL introduces extended pointers (XPointers) to provide a sophisticated method of locating resources.

➤ **Style and Substance**

Since XML documents have no fixed tag set, the presentation of an XML document is dependent on a style sheet. There are two different ways to display XML documents, cascading style sheets (CSS) and extensible style language (XSL).

XSL is likely to be focused on DSSSL (the Document Style Semantics and Specification Language). DSSSL is an international standard style sheet language [ISO/IEC 10179:1996].

For more technical details of HTML, XML and SGML, please refer to [Bray 1997], [Bray 1998], [Goldfarb 1998] and [ISO 8879:1986].

3.5.2 Document object model (DOM)

The document object model (DOM), as a W3C (The World Wide Web Consortium) specification, is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in a broad sense. Increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents.

DOM can be used to manage XML data. With the document object model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the document object model.

In the DOM, documents have a logical structure which is very much like a tree. To be more precise, it is like a "forest" or "grove", and can contain more than one tree. Each document contains zero or one node, one root element node, and zero or more comments or processing instructions; the root element serves as the root of the element tree for the document. The information items in the document can be reached by using "tree-walking" methods. For example, the following XML document can be parsed to the DOM tree as in Figure 3.4.

In XML:

```
<vertex_geometry>
  <cartesian_point E-ID="830" name="" representation_item.wrl="unknown"
    geometric_representation_item.wrl="unknown">
    <coordinates>3.500000</coordinates>
    <coordinates>8.790000</coordinates>
    <coordinates>1.470000</coordinates>
    <dim wrl="unknown" />
  </cartesian_point>
</vertex_geometry>
```

DOM tree:

Figure 3.4 DOM tree

However, the DOM does not specify that documents must be implemented as a tree or a grove, nor does it specify how the relationships among objects are to be implemented. The DOM is a logical model that may be implemented in any convenient manner. The DOM is designed to be used with any programming language such as C++, Java.

3.6 Summary

The existing collaborative systems are restricted on one specific CAD system and do not support multiple systems. However, employing various CAD systems to facilitate the product design process is a desideratum. The objective of this research is to investigate and develop the design collaboration mechanisms for an Internet based design system. The targeted system is to enable the active participation of various members in a design team either within an organization or among various organizations. The major requirement of this system is co-modeling using a set of heterogeneous CAD systems in addition to visualizing the product data; the design team members are able to perform geometric modeling and modification simultaneously. A command-based solution is proposed to deal with four issues identified in Chapter 2. *Operation* is used as the basic transmission data unit to realize the collaboration. The mechanisms of collaborative design and communication mode are discussed. The details on operation definition, methods, and implementation are described in Chapter 4.

CHAPTER 4 COLLABORATIVE OPERATIONS FOR DATA SHARING/EXCHANGE WITH HETEROGENEOUS CAD SYSTEMS

4.1 Introduction

As discussed in Chapter 2, for developing a collaborative CAD system, one of the major issues is CAD data sharing/exchange. The sharing/exchange process in a collaborative CAD system should be dynamic, incremental and able to support various CAD systems. The key challenge is the interactions among functions of different systems. In short, a synchronous and dynamic functional interoperability is needed for a collaborative CAD system. To the best knowledge of the author, there is no reported research that can meet these requirements. In this chapter, a new method, based on CAD software operations, is proposed to address the issue.

4.2 Premise — parametric feature modeling

Parametric design technology is one of the core technologies of integrated CAD/CAM systems. Feature technology, as a milestone, has been developed along the historical process. However, this technology requires further integration and automation to cover the whole product life cycles. Feature-based design is to design a product with features that are functionally defined by attributes and are geometrically represented by a set of parameters [Anderson 1990, Rossignac 1990]. Parametric design deals with variable dimensions as controlling parameters and it enables effective model creation and modifications [Roller 1991, Lee 1996]. Parametric feature modeling, the combination of feature modeling and parametric technology, has been able to support many kinds of design modes (e.g. top-

down or bottom-up, etc.) and design forms (initial design, resembling design and variant design, etc.). In addition, it can capture the engineering information and maintain flexible models.

The proposed method is based on parametric feature modeling technology. To simplify and verify this research work, some basic form features are selected as illustrated in Figure 4.1 by comparing some popular commercial CAD software tools such as Unigraphics, Pro/E, Catia and SolidWorks and the design form features defined in STEP [ISO 2000].

Here, one point to be explained is the sketch feature. Strictly speaking, the sketch is a tool that provides convenient means to define and create a feature. It is not a form feature in fact. A sketch includes the attached plane and a two-dimension boundary. By sketching a rough contour of the desired feature, the designer defines the topology of the feature by its edges and vertices. Then by constraining the sketch, some information, such as position, orientation of the desired feature is defined. Thus the sketch is a special feature that contains some information of the form feature it creates. On the other hand, the sketch is also employed by most current commercial CAD systems as a tool to create form feature. So the sketch is selected as a basic feature to support the proposed method for data sharing/exchange.

Figure 4.1 Classification of form features

4.3 Operation in CAD software

As illustrated in Figure 4.2 (a), a traditional CAD system consists of several layers, such as data repository file, runtime database, modeling kernel, application functions and CAD modular applications. The modeling kernel deals with the low-level geometric data and mainly takes charge of creating, maintaining and displaying the geometric model. It provides basic functions for the application layer to access the database and data repository (usually in the form of files). The modular application layer is an intermediate layer which provides functional interfaces and controls the sessions. It acts as a bridge that connects the user functions to the lower level generic application functions and further to the modeling kernel. This layer enables a series of operations for the designer to input the design data and design intent into the CAD system. According to the design intent, the designer chooses a group of operations to create the part. The application layer extracts the information of the operation and converts the operations into one or more functions provided by lower level sub-systems, such as the generic application module and the modeling kernel. During this process, the design data and knowledge are converted and embedded into the lower level CAD data and stored in the repository via the modeling kernel. According to the input, the modeling kernel modifies the geometric model and updates the display.

(a) Traditional CAD structure

(b) Proposed CAD structure

Figure 4.2 Schematic comparison of traditional and proposed CAD structures

The operation in a CAD system is a set of related commands which are provided by software to achieve certain functional manipulations on entities. It is directly used to support the interface of a CAD system. It is a carrier that carries the design data and design intent. Operations provided by a CAD system act as interfacing functions for the designer to access the CAD system data and functions as illustrated in Figure 4.2(b). The designer uses the operations to design the part step by step. During the design process, the designer converts the design concepts and rules into a group of operations. Through these operations, the design data and design intent are created and maintained by the CAD system.

A feature consists of geometry and non-geometry information which is characterized by a set of parameters. Each feature type in a CAD system can be seen as a pre-defined data structure. An operation can be defined on top of a feature. For example, an operation can instance a feature; then the feature is the result of the operation. Nowadays, most CAD software can support feature modeling and parametric design, hence the operations of CAD systems can usually be related to the high-level design tasks that manipulate features.

From the viewpoint of operational functions, the CAD operations can be classified into two types, namely geometry-related and non-geometry related operations, as illustrated in Figure 4.3. A geometry-related operation affects the geometric model; it either creates new objects into the geometric model or modifies the existing objects. The non-geometric operations could be divided into “auxiliary” and “additional” operation groups. An “auxiliary” operation such as that for layer management or view manipulation mainly facilitates the designer in creating a geometric model and does not affect the geometric model. The rest of the operations can be classified as additional operations, such as those for file management.

Figure 4.3 Classification of operations in a CAD system

The geometric operations can be classified into feature-related and low-level operations according to the objects that they create or modify. The low-level operations create or modify low-level entities of a CAD system, such as points, lines and faces. The feature-related operations create or modify features in a CAD system. The feature-related operations of CAD systems usually are only related to the high-level geometric data.

Obviously, the operation must be executed among different CAD systems via the different functions provided. It means that operations initiated by one CAD system can be carried out with some equivalent functions in another CAD system, which is very useful for collaborative sessions. By mapping appropriate operations between different CAD systems, operations are interoperable with equivalent definitions and functions across multiple CAD systems directly via a standardizing scheme. By executing the corresponding operations, the same product model can be generated. Similarly, the screen display can be synchronized by this means.

4.4 Definition of operation

A feature can be viewed as a pre-defined data structure that provides an easy way to manipulate a geometric model, as shown in Figure 4.4. In this research *operations* in the collaborative design layer are the basic elements for collaboration. They sit on top of the feature representation layer, with effective functions and sequence encapsulated. In other words, *operations* are the dynamic processes to manipulate features. Multiple CAD systems can be integrated through functional mapping and high-level *operation* exchange.

Figure 4.4 Elements of collaborative design

The *operation* type has three constituents, namely, 1) functions; 2) design sequence; and 3) target and reference entity IDs. An operation encapsulates enough CAD information to result in the changes of a product model. Since traditional standalone design usually employs features as design elements, as shown in Figure 4.4, the function of a geometry-

related operation is mainly to create a feature or to modify an existing feature. Thus the CAD information encapsulated in operations is usually high-level semantic messages to instantiate a feature or to modify a feature. The sequence of an operation is generated naturally during the design process which makes the operation a natural unit. This makes the operation have a strong time characteristic. Compared to features, operation is more suitable to be the transmission data unit for a collaborative application. In contrast to the traditional feature definitions [Shah 1989], in this work, a feature is defined as an associative constituent of a product model [Chen 2004a] across different domains of engineering; in design aspect, it provides an easy way for a designer to manipulate the geometric model. Feature does not possess time characteristic and is not sufficient to overcome the shortcomings of existing data exchange strategies as discussed in Section 2.5.1. The base operation type is defined as a generic data type, and is represented in Figure 4.5.

Base operation class
Operation type: featrue related, low-level related, auxiliary, additional
Operation attributes: name, time stamp, sequence
Operation id: int
Target object: taget_object_list
Reference object: ref_object_list
Operation parameter: parameters
Check_validity() Run_Function () Querv_information ()

Figure 4.5 The class definition for operation objects

The type of operation refers to the classification of operations as shown in Figure 4.3. The parameters of an operation vary depending on its type. For example, if the operation is feature-related, the parameters will be the information related to a feature, such as feature parameters, location, orientation and constraint. The target objects are those affected by the operation. The reference objects provide the required information for the operation. The method `Check_validity` is to check if an operation is executable. The individual actual operation can be derived from this base operation class. Figure 4.6 shows the hierarchical structure of operation sub-classes. Here the sketch operation is used to encapsulate the low-level sketch related operations, such as points and lines, to improve the efficiency and effectiveness of collaborative design.

Figure 4.6 Operation class hierarchy

From the discussion above, a conclusion can be drawn that *operation* can be used as a basic transmission data unit to realize dynamic CAD data sharing/exchange among different CAD systems. Using operation as a basic transmission CAD data unit can achieve the following advantages:

➤ Small size

Nowadays, most CAD systems can support feature modeling and parametric design, and feature is usually employed as design element. During the process of collaborative design, only high-level semantic messages (such as specifying feature-related operations and data) are transfer between the client and the server. Thus using operation representation can achieve a rather smaller size than other representations, such as original CAD files or STEP files, which can enhance the effectiveness and efficiency of communication in a collaborative design.

➤ Complete information

An operation not only contains the information to create a geometric model, but also carries the design intent of the designer. Using operations as the transmission data unit can realize high-level CAD data sharing/exchange. Furthermore, by transferring non-geometric operations, other kinds of data, such as layer information and screen view can be synchronized among remote users.

➤ Incremental transfer

Operations can only be generated one by one during the design process, which makes operations have a strong time characteristic. For the development of a collaborative CAD system, this strong time characteristic is very useful. In a collaborative environment, when one user wants to do some modifications, he must select one or more operations provided by the CAD system to implement his intent. After finishing modifications, by propagating the operations to remote users and executing the operations, the same modifications can be

achieved on the remote sites. This way, a nearly real-time co-design can be realized. Thus using operations can support nearly real time incremental co-modeling.

➤ Multiple CAD systems supported

Operation is a special way to represent CAD data and it rarely deals with the low-level geometric data. As discussed above, by applying appropriate modifications on the operation, it can be executed by different CAD systems. Compared to complicated geometric data conversion for exchanging, operation level interoperability is more feasible.

4.5 Requirement of standard operation

Since different CAD systems may have different ways to achieve an operation, an operation conversion mechanism is needed to address CAD functional interoperability. In the proposed solution, a standard operation library is used to construct a channel for CAD data sharing/exchange among various CAD systems.

The requirements for the standard operation are as follows:

- Not tied to a single vendor software solution;
- Simple and generic;
- Independent of any one product development process;
- Capable of implementing common CAD modeling functions; and
- Extensible to create broader engineering applications.

The standard operation also can be used to address the issues of synchronization and consistency. As discussed in Section 2.9.3, to address the issue of synchronization and consistency two aspects need to be considered. Generally speaking, the first aspect is to determine whether the geometric models on each collaborative site are synchronized and consistent. The second aspect is how to synchronize them and maintains them consistent if one geometric model is out of date.

The standard operations can be employed as the neutral validation criterion. By comparing the standard operations recorded on each collaborative site, the synchronization of the geometric models can be determined. By maintaining the consistency of the standard operations, the consistency of differently formatted geometric models is achieved. This capability is based on the assumption that all the features of the CAD systems involved can be mapped into the equivalent ones via a feature-level standardizing scheme.

4.6 Naming mechanism

During the process of collaborative design, when one designer picks up an entity and does some modifications, on the other collaborative site, the same entity must be identified so that the same modifications can be implemented in order to maintain the consistency of the CAD model. How to name these entities is a problem called naming consistency in the development of the collaborative CAD system. This problem becomes more complex because of different data representations among different CAD systems.

The entities of a geometric model can be classified into high-level entities, such as features, and low-level entities, such as solids, faces, lines and points. According to this classification, the problem of naming consistency is divided into two sub-problems, one is feature naming consistency and the other is low-level entity naming consistency.

- Feature naming consistency

Features are the target, result or reference entities of operations and there are associative mapping relations between the features and the operations. Such a relationship can be used to name features among different CAD systems although the features provided in different CAD systems may be different.

As illustrated in figure 3.2, to support various CAD systems, local operations need to be mapped into standard operations. An operation-mapping table is constructed to record the

connections between local operations and standard operations. At the same time, during the execution of local operations, according to the mapping relationship between local operations and local features, another mapping table can be constructed. By combining these two mapping tables, a virtual feature-mapping table for features of different CAD systems is constructed, as illustrated in Figure 4.7. This virtual feature-mapping table can be dynamically created and maintained throughout the lifecycle of a collaborative design process.

Figure 4.7 Virtual feature mapping table

By the virtual mapping table, the virtual mapping relationship of features among different CAD systems can be created and maintained. The virtual mapping relationship can be used to address the naming consistency for features.

- Low-level entity naming consistency

To express the design intent, especially the parameterization information, it is inevitable that operations contain some low-level geometric entities. To execute the operation in another CAD system, these entities must be identified in that CAD system.

The basic idea to address this sub-problem is to pre-define these low-level entities in a neutral format. The entity is defined by its basic attributes by which the entity can be named in any CAD system. For example, an edge can be defined by two vertices and by the two vertices the same edge can be named in various CAD systems.

According to the features defined in Section 4.2, some relative low-level entities are defined as follows:

- o Coordinate system

Coordinate system is the basic entity for CAD and provides the reference for other entities. In the proposed method, a Cartesian coordinate system is defined as the absolute coordinate system and all the other entities are measured by this absolute coordinate system.

- o Point

In the proposed method, a point can be an individual point or an element of other entity. For example, a point can be a vertex of a line or the center of a circle. A point is defined thus:

$$Point_{ij}(x, y, z);$$

i : represents the point is created in i th operation;

j : represents the point is the j th point in the operation.

x, y, z : represents the location of the point

By subscript i & j , different points can be distinguished and by the coordinate value x, y, z , the point can be named in any CAD system.

- o Line

In the proposed method, a line is defined by its two vertices and its format is as follows:

$Line_{ij} (Point_{ik} (x,y,z), Point_{il} (x,y,z));$

i : represents the line is created in i th operation;

j : represents the line is j th line in the operation;

$Point_{ik} (x,y,z), Point_{il} (x,y,z)$: represents the two vertices of the line.

- o Circle

A circle can be represented by its radius and its center point. In the proposed method, a circle is defined as:

$Circle_{ij} (radius, Point_{ik} (x,y,z))$

i : represents the circle is created in i th operation;

j : represents the circle is j th circle in the operation;

$radius$: represents the radius of the circle;

$Point_{ik} (x,y,z)$: represents the center point of the circle.

- o Arc

An arc can be represented by two vertices in counterclockwise direction and the centre point. In the proposed method, an arc is defined as:

$Arc_{ij} (Point_{ik}(x,y,z) , Point_{il} (x,y,z) , Point_{im} (x,y,z));$

i : represents the arc is created in i th operation;

j : represents the arc is j th arc in the operation;

$Point_{ik} (x,y,z) , Point_{il} (x,y,z)$: represents the two vertices of the arc;

$Point_{im}(x,y,z)$: represents the center point of the arc.

- o Edge

In the proposed method, there are two types of edge, namely linear edge and circular edge. For a linear edge, it can be represented by two vertices like a line and is defined as:

$$\text{Edge_linear}_{ij}(\text{Point}_{ik}(x,y,z), \text{Point}_{il}(x,y,z));$$

i : represents the edge is created in i th operation;

j : represents the edge is j th edge in the operation;

$\text{Point}_{ik}(x,y,z), \text{Point}_{il}(x,y,z)$: represents the two vertices of the edge.

For a circular edge, it can be represented by two vertices in counterclockwise direction and its center point. If the two vertices are same, the edge is a full circle. A circular edge is defined as:

$$\text{Edge_circular}_{ij}(\text{Point}_{ik}(x,y,z), \text{Point}_{il}(x,y,z), \text{Point}_{im}(x,y,z));$$

i : represents the edge is created in i th operation;

j : represents the edge is j th edge in the operation;

$\text{Point}_{ik}(x,y,z), \text{Point}_{il}(x,y,z)$: represents the two vertices of the edge;

$\text{Point}_{im}(x,y,z)$: represents the center point of the edge.

o Face

In the proposed method, the face is mainly used to provide an attached face for features. Thus in the proposed method, only the planar face is used. For a planar face, it can be represented by any point on it and its normal. A face can be defined as:

$$\text{Face}_{ij}(\text{Point}_{ik}(x,y,z), V(x,y,z));$$

i : represents the face is created in i th operation;

j : represents the face is j th face in the operation;

$\text{Point}_{ik}(x,y,z)$: represents a point on the face;

$V(x,y,z)$: represents normal of the face.

o Plane

A plane can be defined the same with a face as:

$Plane_{ij}(Point_{ik}(x,y,z), V(x,y,z))$;

i : represents the plane is created in i th operation;

j : represents the plane is j th plane in the operation;

$Point_{ik}(x,y,z)$: represents a point on the plane;

$V(x,y,z)$: represents normal of the plane.

4.7 Conclusions

Using *operations* as the collaborative CAD transmission unit can achieve the following advantages:

1. Reduction in the size of data to be transmitted;
2. Support for incremental modeling;
3. Transferring of complete information;
4. Nearly real time CAD collaboration;
5. Interoperability at the feature level to support various CAD systems.

CHAPTER 5 DESIGN OF A COLLABORATIVE SYSTEM

5.1 Overview

In this chapter, the architecture of a collaborative CAD system is proposed. Detailed functional modules in this architecture are explained. In addition, the implementation of the proposed system is illustrated with a case study.

5.2 System design

5.2.1 System requirements

The requirements of distributed collaborative engineering systems are as follows:

- ❑ Enabling network communication. A collaborative CAD system must provide an environment for a virtual team. This environment should enable remote designers to work together and to communicate among themselves on a common design activity.
- ❑ Sharing high-level CAD data. To enable collaboration among remote users effectively and efficiently, a collaborative CAD must support sharing not only low-level geometric model data, but also high-level design information such as features, constraints and parameters.
- ❑ Supporting feature-level interoperability. As user-preferred semantic type, a mechanism is needed to enable feature-level communication among different CAD packages.
- ❑ Maintaining consistency. It should provide effective concurrency & synchronization control to support multi-user applications.

5.2.2 System framework

The web-based, feature-oriented system architecture [Chen 2005] has been proposed according to the system requirements to enable the collaborative design, as shown in Figure 5.1. The modeling-client/communication-server architecture is adopted. The modeling functions are carried out on individual client with the different CAD systems, and the server mainly provides a communication platform. As for run-time architecture, the replicated architecture is adopted in the proposed system. Each client maintains local format CAD file and the flexible mechanism called administrator/controller discussed in Chapter 3 is developed to coordinate collaborative design sessions. For network communication, the most popular communication and network protocol, TCP/IP, is adopted.

5.2.3 Function analysis of the system

➤ Client

The CAD system on the client is mainly responsible for carrying out modeling functions. A network module is embedded into the selected CAD system based on its API to extend the traditional single-location CAD system to a multi-location application. The functional modules on the client's side are discussed in more detail as follows:

- CAD system

Each client CAD system could be any current CAD system as long as it meets the requirements of the client CAD system discussed in Chapter 3. This way, the collaborative system can make use of the strong modeling functions from the selected CAD systems.

Figure 5.1 System architecture

- Local operation manager

The local operation manager is an application program based on the API that the CAD system provides. It has two major functions. First, it acts as an intermediate layer between the CAD system and the proposed collaborative system and is responsible for exchanging operations. Second, it is responsible for maintaining the local operation & feature mapping table. In the mapping table, the mapping relationships between the features in the local geometric model and the operations are recorded. The parameters of the operations are also stored in this table.

When the control right is assigned to the client, features can be created and existing features can be modified. If a feature is created, the local operation manager is responsible for extracting operations from the CAD system and adding the operation and feature mapping relationship into the local operation and feature mapping table. If an existing feature is modified, the local operation manager extracts the modification information from the CAD system, and then the corresponding operation from the mapping table is retrieved and updated.

When the client is in observing status, the client only can execute the operation from other clients to update the model on his/her machine. The local operation manager mainly takes charge of executing the operations on the local CAD system. Similarly, if an operation from other clients is to create a feature, a feature creation local operation will be executed and a new mapping relationship will be added into the mapping table on this client's machine. If an operation is to modify the existing feature, the local operation manager will retrieve the corresponding feature by operation according to the mapping table and update the parameters of the operation in the table. Then the new operation for generating the same modification on the local geometric model will be executed on the local CAD system.

- Operation mapping module

The operation mapping module is responsible for mapping local operations into standard operations for CAD interoperability. The mapping table between local and standard operations is dynamically created and maintained to record the mapping relationship and store all the parameters of standard operations.

- Communication module

The communication module on the client's side is responsible for exchanging information with the server. It contains a process listening to incoming information continuously from the socket connected to the communication module on the server. The outgoing information is sent via it, too. Two types of information will be exchanged. One is the standard operation that conveys the design operations and design intent. The other is the management data that is related to the collaborative design session management.

The client-side communication module consists of three parts.

The first part provides collaborative session management service. This part is incorporated into the CAD system based on its API to provide interfaces for managing a collaborative session, such as starting and ending a new collaborative session, and exchanging information on geometric model status. Also the functions of the administrator/controller mechanism are provided if the client is an administrator.

The second part is responsible for encoding or decoding XML-format information.

The third part mainly provides network communication functions. This part is an application program that is independent of the API of the CAD system because usually it does not provide network functions. TCP/IP socket connection is used to set up the communication channel between clients and server.

➤ Server

In the proposed system, the server mainly acts as the information exchanging centre. All the data and information that originate from a client will be transmitted to the server where they are recorded, managed, controlled and dispatched to other clients connected to the server. Also the server takes certain responsibility for coordinating the collaborative session.

Since the modeling functions are carried out by the CAD system on each client's machine, the functionality related to modeling such as querying information of entity, updating display can be fulfilled from the client's side and does not need the support of the server. This will greatly reduce network load and the amount of data exchanged between the server and the clients. Therefore, this design has the advantage of avoiding the server from becoming the bottleneck of performance.

The functional modules on the server's side are discussed as follows:

- Communication module

Similar to the communication module on the client, the communication module on the server constructs a communication platform of whole system to support the collaborative design. But, unlike the communication module on the client, the server-side communication module is only responsible for receiving data from the clients, encoding or decoding XML formatted information, and dispatching the data to the designated clients.

Presently, computers running on the Internet communicate to each other using either the Transport Control Protocol (TCP) or the User Datagram Protocol (UDP). UDP is a protocol that sends independent packets of data, called datagram, from one computer to another with no guarantees about arrival. In the proposed system the connection-based protocol, TCP/IP, is selected, as it is more reliable.

- Session manager

The session manager has two functions. One is to maintain a socket connection for each client. Once it receives information from a client, it either processes the information itself if possible, or, propagates it to the other clients.

The second function is to coordinate the collaborative design sessions. It assists the communication module on the client's side to implement the administrator/controller mechanism.

- Standard operation manager

The standard operation manager is responsible for creating and maintaining the standard operation table on the server's side. The standard operation table is to store the standard operations and their parameters. The standard operations and their parameters can be seen as the central geometric model that is maintained on the server, although it is not really a CAD geometric model.

The standard operation manager handles all the operations from clients. It dynamically updates the standard operation table in order to keep the latest results. As discussed previously, these server-side standard operations and client-side standard operations can be used to maintain consistency of differently formatted geometric models.

5.3 Case study

Based on the proposed system architecture, a collaborative CAD prototype system has been developed to verify the feasibility of using *operation* as the basic transmission data unit to realize dynamic CAD data sharing/exchange among different CAD systems. Two commercial CAD systems, Unigraphics and SolidWorks, are selected to carry out modeling functions on the client's side. The prototype system employs features as design

elements, thus only those operations which create or modify features, are shared and communicated.

5.3.1 Standard operation

To address CAD interoperability, a set of standard operations are pre-defined as illustrated in Figure 5.2 based on the classification of the design form features as illustrated in Figure 4.1. These operations are all geometry-related ones.

Figure 5.2 Standard operations

As illustrated in Figure 5.3, besides the standard operations, a set of mapping rules are defined for Unigraphics and SolidWorks respectively to map the local operations of selected CAD systems to the standard operations.

Figure 5.3 Operations mapping table between Unigraphics and SolidWorks

Different CAD systems may have different geometric coverage. Each CAD system may have its own strong point and special modeling functions. The geometric modeling

functions of a CAD system can be classified into two categories, namely non-free-form and free-form modeling. Here non-free-form modeling refers to those functions to create solids whose faces can be defined by an equation, e.g. $z = f(x, y)$, such as flat, cylindrical and conic faces. Free-form modeling refers to those functions to create solids or surfaces that contain free-form surfaces, such as Bézier, B-spline and NURBS (Non-Uniform Rational B-Splines) surfaces. After a detailed study on the modeling functions of some popular commercial CAD software tools such as Unigraphics, Pro/E, Catia and SolidWorks, it can be found that for non-free-form modeling, these CAD tools have almost the same coverage. The main differences lie in the field of free-form modeling. The definitions, representations, construction methods of free-form surface differ greatly. The underlying techniques of these commercial CAD tools are not clear due to commercial protection. At this stage, the author mainly focuses on the non-free-form modeling. The standard operations listed in Figure 5.2 are all non-free-form modeling related operations. As to free-form geometry interoperability, more research work has to be done.

5.3.2 User interface

To enable the collaborative design, user interfaces are developed on the selected CAD system based on its API. The main user interfaces for Unigraphics and SolidWorks are shown as in Figure 5.4 and 5.5.

Figure 5.4 The main interface for Unigraphics

Figure 5.5 The main interface for SolidWorks

5.3.3 Demonstration

This section demonstrates the use of prototype system in supporting the collaborative design with Unigraphics and SolidWorks through a simple scenario. In the scenario, three collaborating designers are assumed to collaboratively build a part. Generally, there are mainly two functions to test for evaluation of this prototype system. One is the modeling functions, and the other is the network functions. Figure 5.3 lists all the modeling functions supported by the prototype system. Many parts have been tested on the prototype system successfully. Two typical parts are selected to illustrate the collaborative modeling process. Some successfully tested parts are also listed in Appendix C.

5.3.3.1 Testing part 1

In this section, a typical plastic part is used to test the prototype system as shown in Figure 5.6. One of possible modeling processes is shown in Figure 5.7 based on the standard operations.

Figure 5.6 Demonstration part 1

Figure 5.7 The modeling process

➤ Scenario 1: initializing the collaborative design space

Assume that at the beginning, there are two users to design the part collaboratively, named site 0 with Unigraphics, and site 1 with SolidWorks. At first, Site 0 activates the server program to set up a collaborative design space as shown in Figure 5.8. The server program is an independent executable program that can be located inside or outside of the user’s machine. And site 0 becomes the administrator.

Figure 5.8 A message about the server

Next site 0 and site 1 can click the “start co-design” button to join the collaborative design. A pop-up dialog will display for users to log on the server as shown in Figure 5.9(a) and 5.10(a). Figure 5.9(b) and Figure 5.10(b) show the main interfaces after users log on (here site 0 logs on first). Figure 5.11 shows the status of server after users log on.

Figure 5.9 User log on interface within Unigraphics

Figure 5.10 User log on interface within SolidWorks

Figure 5.11 Screen messages reflected on the server

➤ Scenario 2: collaboratively modeling the part

After logging on to the server, the collaborative design can be started. At the first round, the site 0 clicks the “Request” button to get the right to edit the model. The server processes the incoming requests and assigns the control baton to site 0 according to the principle of “first come, first serve” and informs site 1 to wait. After getting the control

baton, site 0 becomes the controller as shown in Figure 5.12 and has the right to edit the model. Site 1 is the viewer and only can view the model.

Figure 5.12 The screen layout of the co-design interface at (a) site 0, and (b) site 1

Site 0 uses the “Block” operation to create the base feature in Unigraphics. After finishing the operation, he clicks the “SendOperation” button, and the “Block” operation of Unigraphics is mapping into standard operation “Block”. First, the local operation manager on site 0 will invoke the Unigraphics API function *UF_MODL_ask_block_parms* to extract the information of the block feature. This information is used to constitute the local operation. After that, the operation mapping module will map the local operation into standard operation “Block”. The standard operation with the control baton is sent to the server. The standard operation in this case is shown as follows:

```
operation_name: BLOCK(0)
operation_parameter:
    p0=100
    p1=70
```


$p2=15$

location:

$X=0.000000$

$Y=0.000000$

$Z=0.000000$

#

In the standard operation “Block”, $p0$, $p1$, $p2$ mean the length, width and height of the block feature, respectively. X , Y and Z are the coordinates of the corner point of the block feature. For the block feature, the orientation is defined as the same as the coordinate system. Namely, the direction of length, width and height are the same as the direction of X , Y and Z axis of the coordinate system, respectively. So the standard operation “Block” has no orientation information. The information in the standard operation “Block” is rather simple but enough to create a block feature. This is because the operation is defined based on the feature. The size of this standard operation “Block” is only 0.2 KB. If using the STEP or IGS, to transfer the information of this block shape, the size of the transmitted data between the client and the server is about 8 KB and 20 KB respectively. Clearly, by sending operations, the communication load between server and distributed clients can be minimized.

The server delivers the standard operation to site 1 and sends a confirmation to site 0. Upon receiving the standard operation, site 1 clicks the “Update” button to update the local model on his/her machine. First, the standard operation is mapped into the local operation by the operation mapping module. Since in SolidWorks there is no block feature, the standard operation “Block” is mapping into the local operation “Extrude_base”. During the mapping, the information in standard operation “Block” is converted to three types of information to constitute the local operation “Extrude_base”. The first is the attached plane. This attached plane is used to create a sketch for extrusion in SolidWorks. According to the

standard operation, it is the XY plane and can be defined by a point on it such as point (0, 0, 0) and its normal, namely Z direction (0, 0, 1). The second is the sketch. According to the standard operation, the sketch includes a rectangular which has the same length and width with the block feature. The third is the extrusion parameters. According to the standard operation, there is only one parameter, namely the extrusion height which is the same with the height of the block feature. After mapping, the local operation manager will invoke the API functions of SolidWorks to execute this local operation to update the model. First, it will invoke the function *LPENTITY->Select* or *LPMODELDOC->ICreatePlane* to get the attached plane. Then it will invoke the *LPSKETCH->IGetActiveSketch* to initialize a sketch and *LPMODELDOC->ICreateLine* to create the rectangular. After that, it will invoke *LPPARTDOC->FeatureExtrusion* to create the extrusion. After successfully updating local model, an update confirmation is then sent to the server. After receiving the update confirmation, the server informs all sites to start a new round of collaborative design (see Figure 5.13).

(a)

(b)

(c)

Figure 5.13 The screen layout of the co-design interface at (a) site 0, and (b) site 1 and (c) server

At second round, site 1 clicks the “Request” button first and gets the control baton. This time, site 1 is the controller and site 0 is the viewer. Site 1 uses the “Extrude_boss” operation to create second feature in SolidWorks. After finishing the operation, he clicks the “SendOperation” button, and the “Extrude_boss” operation of SolidWorks is mapping into standard operation “Pad”. First, the local operation manager on site 1 will invoke the SolidWorks API function *LPUNKNOWN-> QueryInterface* to extract the information of the “Extrude_boss” feature. This information is used to constitute the local operation. According to the shape of the extrusion profile, the local operation “Extrude_boss” of SolidWorks can be mapped into the standard operation “Boss”, “Pad” or “Extrude_add”. In this case, it is mapped into the standard operation “Pad” because its extrusion profile is a rectangular. The standard operation in this case is shown below:

The size of this standard operation “Pad” is only 0.4 KB. If using the STEP or IGS, to transfer the information of the current geometric model, the size of the transmitted data between the client and the server is about 11 KB and 30 KB respectively.

Upon receiving the standard operation, site 0 clicks the “Update” button to update the local model. First, the standard operation is mapped into local operation by the operation mapping module. During the mapping, the information in the standard operation “Pad” is converted to three types of information to constitute the local operation “Pad”. The first is the attached plane. This attached plane is used to attach the pad feature in Unigraphics. According to the standard operation, it is the top face of the block feature and can be defined by a point on it such as point (0, 0, 15) and its normal, namely Z direction (0, 0, 1). The second is the parameters of the pad feature. According to the standard operation, there are length, width and height of the pad feature. The third part is the orientation of the pad feature. According to the standard operation, the direction of the length of the pad feature is along to X direction of the coordinate system and can be defined as (1, 0, 0). After mapping, the local operation manager will invoke the API functions of Unigraphics to execute this local operation to update the model. First, it will invoke the function *UF_OBJ_ask_type_and_subtype* to get the attached plane or *UF_MODL_create_plane* to create the attached plane if the attached plane is not an exiting face of the current solid model. Then it will invoke the *UF_MODL_create_rect_pad* to create the pad feature. After successfully updating local model, the geometric model on the machine of site 0 will be the same with the one on the machine of site 1 as shown in Figure 5.14.

(a)

(b)

```
Shortcut to server.exe
message from 116:operation_name: BLOCK<0>
operation_parameter:
  p0=100
  p1=70
  p2=15
location:
  X=0.000000
  Y=0.000000
  Z=0.000000
#
message from 116:CONTROL MESSAGE:FINISH SENDING MESSAGE
message from 136:CONTROL MESSAGE: REQUEST1?
message from 116:CONTROL MESSAGE: INQUIRY THE STATUS OF SERVER
message from 136:operation_name: Pad
operation_parameter:
  X      : p3=60
  Y      : p4=70
  Z      : p5=15
  Corner_rad : p6=0
  Taper_angel:p7=0
feature_direction
direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=1.000000
X direction
  dir_y[0]=1.000000
  dir_y[1]=0.000000
  dir_y[2]=0.000000
location:
  X=0.000000
  Y=0.000000
  Z=15.000000
#
message from 136:CONTROL MESSAGE:FINISH SENDING MESSAGE
```

(c)

Figure 5.14 The screen layout of the co-design interface at (a) site 0, and (b) site 1 and (c) server

At any time, a third user can join the collaborative design. He clicks the “start co-design” button to send a request to the server. The server will wait until the controller sends the latest standard operation to it. After receiving the latest standard operation, the server will send the previous standard operations and the latest standard operation to site 2. And site 2 can request the control baton at the next round of collaborative design.

By repeating the process described as above, the targeted part is collaboratively designed. If a user is satisfied with design, he clicks the “Satisfy” button. After all the users are satisfied with the part, the server will end the collaborative design session. On each

user’s machine, the same part, as shown in Figure 5.15, is generated although it is stored in different formats. The parts on different machines not only have the same geometric model, but also the same parametric information. After the collaborative design, the size of the standard operations recorded on the server is only 42 KB, while the size of the finished part is 188 KB in STEP format or 764 KB in IGES format. For the detailed standard operations transmitted during the collaborative design, please refer to Appendix A.

(a)

(b)

Figure 5.15 The screen layout of the co-design interface at (a) site 0, and (b) site 1

5.3.3.2 Testing part 2

The part and the operations that may be used to model the part are shown in Figure 5.16. And the corresponding standard operations are also shown. One of possible modeling processes is shown in Figure 5.17 based on the standard operations.

After the collaborative design, the size of the standard operations recorded on the server is only 20 KB, while the size of the finished part is 283 KB in STEP format or 1114 KB in IGES format. For the detailed standard operations transmitted during the collaborative design, please refer to Appendix B.

Figure 5.16 The part and operations

Figure 5.17 The modeling process

5.4 Discussion

From the above demonstration, a nearly real-time collaborative design has been realized. The experimental results from the current prototype system prove that using *operation* as the basic transmission data unit is a feasible way to realize dynamic CAD data sharing/exchange among various CAD systems. This way, not only the geometric data can be exchanged, but also the high-level design information (i.e. features, parameters and geometric constraints) is exchanged, which overcomes the drawback of the traditional data exchange strategies and the standards mentioned in Chapter 2. Compared with other collaborative design solutions discussed in Chapter 2, e.g. CMS [Chan 1999], WPDSS [Qiang 2001], webspIFF [Rafael 2001], this proposed solution has advantages as follows:

➤ High communication efficiency

As shown in the demonstration, the size of the whole standard operations for testing part 1 and testing part 2 is 42 KB and 20 KB respectively. However, in STEP format, it will be 188 KB and 764 KB respectively; in IGES format, it will be 283 KB and 1114 KB respectively. Furthermore, the proposed solution supports the incremental data transfer and the size of single standard operation is usually less than 1 KB. Clearly, by sending operations, the communication efficiency between server and distributed clients has been improved greatly.

➤ Complete feature information

The case study has shown that with the prototype system, the collaborative counterparts share not only the same geometric model, but also the full feature information. This characteristic will benefit the designer greatly and reduce the design cycle time. For example, if there is a modification to remove the “shut off hole” of the testing part 1 as shown in Figure 5.6, it will be very easy using the prototype system. Since the full feature information is retained on all machines of the collaborative designers, the designer either using Unigraphics or SolidWorks just needs to suppress or delete the features that are related to the “shut off hole”. With other collaborative solutions, since the feature information is usually lost, the designer needs additional modeling functions to remove it. This process is usually time-consuming and error-prone.

➤ Multiple CAD systems supported

With the globalization of multinational corporations, product development teams nowadays involve multiple parties, who are often geographically distributed and using different CAD systems. Thus, an environment that supports collaborative design with current popular CAD systems is a desideratum. To the best knowledge of the author, there is no reported research that can support co-design with multiple CAD systems.

➤ Strong modeling functions

The implementation of the proposed method involves only integration work; therefore, the powerful functions of the selected CAD systems are inherited and can greatly benefit client users.

Of course, there are some shortcomings on the prototype system, which enlighten the direction of next stage research.

➤ Operation optimization

The proposed solution is based on the operations. During the collaborative design, what operations are selected and what order the operations may be depends on the modeling experience of the designer. Of course, different options may get the same geometric model. However, the option will affect the stability of the geometric model. It is a common experience that during the design changing one parameter sometimes results in collapse of whole geometric model. The proposed solution has the potential capability to resolve this issue. As mentioned in Section 5.2.3, the standard operations and their parameters can be seen as the central geometric model that is maintained on the server. During the design, the operation will affect the geometric model. After design, the standard operations stored on the server can be optimized according to the geometric model and the design intent. This optimization process is similar with the process of feature recognition. In the future, a set of operation optimization rules will be set up to optimize the standard operations stored on the server after the collaborative design to improve the stability of the geometric model.

➤ Operation mapping

Among different CAD systems, the operation mapping can be one-to-one, multiple-to-one, or multiple-to-multiple. For example, the operation “counter-bore hole” in Unigraphics can be mapped into two “Extrude_cut” operations in SolidWorks.

At this stage, the operation mapping in the prototype system as shown in Figure 5.3 is one-to-one. This has certain limitation on the flexibility of geometric modeling. To provide a more flexible modeling method, and to make full use of the modeling functions of the select CAD systems, more complicated operation mapping methods will be established.

➤ Modeling functions

As mentioned in Section 5.3.1, the standard operations are selected based on the design form features of STEP and only can support non-free-form modeling. The modeling functions of the prototype system need to be extended to take full advantage of powerful modeling functions of select CAD systems.

Furthermore, based on the requirement of modeling functions, a minimum set of standard operations will be established to relieve the development work. The method to provide generic user-defined standard operations and a way to create mapping rules dynamically will be studied to maintain the flexibility of geometric modeling. The concept of user-defined operations is similar to the one of user-define features which are already provided by some commercial CAD systems to enhance the modeling functions and flexibility.

➤ Exception treatment

Some exceptional situations need to be considered to avoid unreasonable monopoly of the control and deadlocks. For example, if the administrator is offline accidentally during the collaborative session, the whole system is under the threat of deadlock.

➤ Real-time collaboration

The current prototype system can only achieve a nearly real-time interactive collaborative design. A mechanism to automatically capture and transfer operations will be investigated to address this problem.

CHAPTER 6 SUMMARY, CONCLUSION & FUTURE WORK

6.1 Summary

This thesis has proposed an approach for the development of a collaborative CAD system. Based on comprehensive survey of literature on the existing collaborative systems, it can be stated that to the best knowledge of the author, the technology that supports multiple CAD tools co-modeling has not been reported in the past.

- The existing collaborative systems are restricted on a specific CAD system and cannot support collaborative design with heterogeneous systems. This research extends the research field of collaborative design from a specific CAD system to multiple ones.
- The key issues to develop a collaborative system supporting multiple CAD systems are investigated. They are feature-level interoperability, data transfer, synchronization and consistency, and concurrency. A novel, command-based and integrated solution is proposed to address these issues.
- The architecture of a collaborative CAD system was proposed. In addition, a prototype system was developed based on two commercial CAD systems, Unigraphics and SolidWorks. This prototype system proves the feasibility of the proposed method that uses *operation* as the basic transmission data unit to address the feature-level interoperability and to realize dynamic high-level CAD data sharing/exchange among different CAD systems.

6.2 Conclusion

Operation is a basic type of information element for collaboration, on top of feature, with effectively operational functions and sequence encapsulated. With the *operations* standardized, multiple CAD systems can be integrated through functional mapping and high-level data sharing/exchange. So far no existing research work is available for operation-based solutions. The author has explored the key mechanisms within a prototype system. The experimental results from the current prototype system have proven the feasibility of the novel method that uses *operation* as the basic transmission data unit to address feature level interoperability and to realize dynamic high-level CAD data sharing/exchange among different CAD systems.

6.3 Future work

Based on the experience of this research, future work is suggested as follows:

- *Standard operation extension*

From the experimental results of the current prototype system, it can be seen that, to make full use of modeling functions of selected CAD systems, a set of standard operations need to be developed and implemented. Such standard operations should be generic and complete in order to meet the requirements of standard operations as discussed in Chapter 4. The following aspects need to be studied:

- I. To investigate and classify CAD entities and the corresponding representations.
- II. To identify the similarity and dissimilarity for the methods in order to unify CAD operations to be provided for different popular CAD systems.
- III. To investigate and specify the generic product development activities and their corresponding generic operations.

- IV. Based on STEP [ISO 2000], to construct the identified generic standard operations and develop the data representations for standard operations.
- V. Based on the requirement of modeling function, to establish a minimum set of standard operations to relieve the development work.
- VI. To study the method to provide user-define standard operations and the way to create mapping rules dynamically.

- *Real-time collaborative design*

The current prototype system can only achieve a nearly real-time collaborative design. It still has a long way to realize real-time collaborative design. A mechanism to automatically capture and transfer operations should be investigated to realize real-time collaborative design.

- *Enhancement of interactivity of the proposed system*

For a successful collaborative CAD system, it should provide a good level of interactivity. In addition, users will not endorse the systems if they do not have the acceptable performance, e.g. a reasonable short delay for every operation.

To enhance the interactivity of the proposed system, a more flexible feature level concurrency control system should be investigated based on the proposed administrator/controller mechanism.

- *Operation optimization and operation mapping rules*

As mentioned in Section 5.4, a set of operation optimization rules will be set up to optimize the standard operations stored on the server after the collaborative design to improve the stability of geometric model and more complicated operation mapping methods will be established to provide a more flexible modeling method, and to make full use of the modeling functions of the select CAD systems.

REFERENCES

- [Adaptive 2006] Adaptive Media Envision3D, Adaptive Media Inc., <http://www.adaptivemedia.com> (Accessed in December 2006).
- [Alibre 2001] Alibre Design Alibre Inc., www.alibre.com, 2001 (Accessed in September 2005).
- [Anderson 1990] Anderson D. C., Chang T.C., “Geometric reasoning in feature-based design and process planning”, *Computer and Graphics*, Vol. 14, No. 2, pp. 225-235, 1993.
- [Actify 2006] Actify SSpinFire, Actify Inc., www.actify.com (Accessed in December 2006).
- [Begole 1997] Begole J, Struble C. A., Shaffer C. A., “Leveraging Java applets: toward collaboration transparency in Java”, *IEEE Internet Compute*, Vol. 1, No. 2, pp. 57-64, 1997.
- [Bray 1997] T. Bray and S. DeRose, *Extensible Markup Language (XML): Part II. Linking*, World Wide Web consortium Working Draft (Work in Progress), August 1997, Available at <http://www.w3.org/TR/WD-xml-link.html>.
- [Bray 1998] T. Bray and J. Paoli, *Extensible Markup Language (XML) 1.0*, W3C Recommendation, Sperberg- McQueen CM, 1998.
- [Bronsvoort 1993] Bronsvoort W.F., Jansen F.W., “Feature modeling and conversion-key concepts to concurrent engineering”, *Computers in Industry*, Vol. 21, No. 61, pp. 61-86, 1993.
- [Bronsvoort 2004] Bronsvoort W.F., Noort A., “Multiple-view feature modeling for integral product development ”, *Computer-Aided Design*, Vol. 30, No. 10, pp. 929-946, September 1, 2004.

- [Burton 1998] Burton, P., "Technologies & Standards", *Technical Report from the European Commission supported WIDEBEAM project*. http://www.isomatic.co.uk/WBStandards.htm#Peter_Burton (Accessed in September 2005).
- [Chan 1999] Chan Stephen, Wong Martin, Ng. Vincent, "Collaborative solid modeling on the WWW", *Proceedings of the ACM symposium on Applied Computing*, pp. 598-602, 1999.
- [Chen 1997] Y. M. Chen and C. L. Wei, "Computer-aided feature-based design for net shape manufacturing", *Computer Integrated Manufacturing Systems*, Vol. 10, No. 2, pp. 147-164, 1997.
- [Chen 2000] Y. M. Chen, Liang M. W., "Design and implementation of a collaborative engineering information on system for allied concurrent engineering", *International Journal Computer Integrate Manufacture*, Vol. 13, No. 1, pp. 11-30, 2000.
- [Chen 2001] Chen L, Song Z. J., Liavas B., "Exploration of a multi-user collaborative assembly environment on the Internet: a case study", *Proceeding of the ASME Design Technical Conferences and Computers and Information in Engineering Conference*, DETC/CIE-21291, Pittsburgh, Pennsylvania, September 9-12, 2001.
- [Chen 2002] Chen L, Song Z. J., Liavas B., "Master assembly model for real-time multi-user collaborative assembly modeling on the Internet ", *Proceeding of the ASME Design Technical Conferences and Computers and Information in Engineering Conference*, DETC/CIE-34456, Montreal, Canada, September 29-October 2, 2002.
- [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”, accepted by *Computer-Aided Design & Applications*, 2005, CAD’05, 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.
- [Cignoni 1998] Cignoni P., Montani C., Scopigno R., “Comparison of mesh simplification algorithms”, *Computers & Graphics (Pergamon)*, Vol. 22, No. 1, pp. 37-54, 1998.
- [Cimmetry Systems Autovue 2006] Cimmetry system, Cimmetry Systems Inc., www.cimmetry.com (Accessed in December 2006).
- [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.
- [Collab 2001] CollabCAD National Informatics Center, India, 2001. www.collabcad.com (Accessed in September 2005).
- [ConceptWorks 2006] ConceptWorksTM, Inflow Inc., www.inflow-tech.com (Accessed in December 2006).

- [Deitz 1995] Deitz D., 1995, "Infrastructure for integration" *Mechanical Engineering*, Vol. 117, No. 3, pp. 78-80, Mar, 1995.
- [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.
- [Dong 1997] Y. Dong, C. F. Y. Chee, Y. He, A. Goh, "Active database support for STEP/EXPRESS models", *Journal of Intelligent Manufacturing*, Vol. 8, No. 4, pp. 251-261, 1997.
- [Dourish 1996] P. Dourish, "Consistency guarantees: exploiting application semantics for consistency management in a collaboration toolkit", *Proceedings of the ACM Conference on Compute Supported Cooperative Work*, pp. 268-277, 1996.
- [Dumaine 1989] Dumaine, B., "How manager can succeed through speed", *Fortune*, Vol. 119, pp. 54-59, 1989.
- [eDrawing 2006] SolidWorks eDrawing, SolidWorks Inc, www.solidworks.com/edrawings (Accessed in December 2006).
- [Everitt 1988] P. B. Everitt, "Product data exchange specification: RAMP's unique implementation of the PDES specification", Dearborn, Mich., Society of Manufacturing Engineers, 1988.
- [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.
- [Gerhard 2001] Gerhard J. F., Rosen D., Allen J. K., Mistree F., "A distributed product realization environment for design and manufacturing", *ASME Journal of Computer Information Science Engineering*, Vol. 1, No. 3, pp. 235-244, 2001.

- [**Goldfarb 1998**] C. Goldfarb, P. Prescod, *The XML Handbook*, in: C. Goldfarb (Ed.), *Open information Management*, Prentice Hall, Upper Saddle River, NJ, ISBN 0-12-081152-1, 1998.
- [**Goldin 1998**] Goldin D. S., Venneri S. L., Noor A. K., "Beyond incremental change", *IEEE Computer*, Vol. 31, No. 10, pp. 31-9, October, 1998.
- [**Gonzalez 2002**] Gonzalez, F. J. M., 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.
- [**Greenberg 1994**] Greengerg S., Marwood D., "Real time groupware as a distributed system: Concurrency control and its effect on the interface", *Proceedings of the ACM Conference on Compute Supported Cooperative Work*, pp. 207-217, 1994.
- [**Greif 1988**] Greif, Irene, *Computer-Support Cooperative Work: A Book of Readings*, Sanmateo, Calif, Morgan Kaufmann Publishers, 1988.
- [**Griffin 1995**] C. R. Griffin, P. Patel, "The transfer and exchange of engineering information", Printed and published by the IEE, Savoy Place, London WC2R OBL, UK, 1995.
- [**GS-Design 2001**] GS-Design Inc., www.gs-design.com, 2001. (Accessed in September 2005)
- [**Hale 2000**] M. A. Hale, D. N. Mavris, "A Lean-Server Foundation of Collaborative Design", *Advances in Engineering Software*, Vol. 31, No. 8-9, pp. 679-685, 2000.
- [**Harold 1999**] E. R. Harold, "XML Bible", IDG Books Worldwide INC., 1999.
- [**Hoops 2001**] Hoops Stream Toolkit, Hoops Inc., www.hoops3d.com, 2001. (Accessed in September 2005)
- [**Inventor 2001**] Inventor collaboration tools, Autodesk Inc., www.autodesk.com, 2001. (Accessed in September 2005)

- [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 2000] Industrial Automation system 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 N998, December, 2000.
- [ISO/IEC 10179: 1996] *Document Style Semantics and Specification Language (DSSSL)*, ISO/IEC 10179, 1996.
- [ISO 8879:1986] *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML)*, ISO 8879, 1986.
- [Jacquel 2000] Jacquel D., Salmon J., “Design for manufacturability: a feature-based agent-driven approach”, *Proceeding of the institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 214, No. 10, pp. 865-879, 2000.
- [Jha 2000] Kailash Jha, Gurumoorthy B., “Multiple feature interpretation across domains”, *Computers in Industry*, Vol. 42, No. 1, pp. 13-32, 2000.
- [Joe 2000] Y. S. Joe, M. R. Svetan, V. George, “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*, pp 379-403, 2000.
- [Kao 1998] Y.C. Kao, G.C.I. Lin, “Development of a collaborative CAD/CAM system” *The Journal of Robotics and Computer-Integrated Manufacturing*, Vol. 14, No. 1, pp. 55-68, 1998.

- [**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.
- [**Kraker 1995**] De Kraker K. J., Dohmen M., Bronsvort W. F., “Multiple-ways feature conversion to support concurrent engineering”, *Proceedings Solid Modeling*, Third Symposium on Solid Modeling and Applications, pp. 105-114, 1995.
- [**Lee 1996**] J. Y. Lee and K. Kim, “Geometric reasoning for knowledge-based parametric design using graph representation”, *Computer-Aided Design*, Vol. 28, No. 10, pp. 831-841, 1996.
- [**Li 2002**] Li W. D., Ong S. K., Nee A. Y. C., “Establishment of a distributed design environment”, *Proceedings of the Ninth ISPE International Conference on Concurrent Engineering: Research and Applications*, UK, pp. 605–612, 2002.
- [**Li 2004**] Li W. D. , Ong S. K., Fuh J. Y. H., Wong Y. S., Lu Y. Q., Nee A. Y. C., “Feature-based design in a distributed and collaborative environment”, *Computer Aided Design*, Vol. 36, No. 9, pp. 775-797, August, 2004.
- [**Lin 1993**] Lin G. C. I., “Management of advanced manufacturing technology”, *Manufacturing Management Technology Seminar Lecture Notes*, National Cheng Kung University, Tainan, Taiwan, 1993.
- [**Liu 2000**] Liu X. D., “CFACA: component framework for feature-based design and process planning”, *Computer Aided Design*, Vol. 32, No. 7, pp. 397-408, 2000.
- [**Ma 2000**] S. Ma, Y. Marechal and J. L. Coulomb, “Methodology for an implementation of the STEP standard: A Java Prototype”, *IEEE Transactions on Magnetic*, Vol. 36, No. 4, July 2000.
- [**Ma 2003**] Ma, Y. S. and Tong, T., “Associative feature modeling for concurrent engineering integration”, *Computers in Industry*, Vol. 51, pp. 51-71, 2003.

[**Mandviwall 1995**] Nandviwall, Munir, Grillo, Peter, “Teambox: an exploration of collaborative interoperability”, *Proceedings of the Conference on Organizational Computing, Milpitas, Canada*, pp. 347-353, Systems, Aug 13-16, 1995.

[**Mantyla 1995**] Mantyla M., *An introduction to Solid Modeling*, Rockville, Md: Computer Science Press, ISBN: 0881751081, 1995.

[**Matrix Design Central 2006**] MatrixOne Product Lifecycle Management Solutions, MatrixOne Inc., www.matrixone.com (Accessed in December 2006).

[**MatrixOne 2006**] Matrix Design Central, MatrixOne Inc., <http://www.matrixone.com/matrixonesolutions/designercentral.html> (Accessed in December 2006).

[**Mowbray 1995**] T. J. Mowbray and R. Zahavi, *The essential CORBA: systems integration using distributed objects*, New York: Wiley, ISBN: 0471106119, 1995.

[**Nam 1998**] Nam T. J., Wright D. K., “CollIDE: a shared 3D workspace for CAD”, *Proceedings of the 1998 Conference on Network Entities*, Leeds, UK, pp. 389–400, 1998.

[**Nidamarthi 2001**] Nidamarthi S., Allen R. H., Sriram R. D., “Observations from supplementing the traditional design process via Internet-based collaboration tool”, *International Journal Computer Integrate Manufacture*, Vol. 14, No. 14, pp. 95-107, 2001.

[**OneSpace 2000**] OneSpace, CoCreate Inc., www.onespace.net, 2000. (Accessed in September 2005)

[**Owen 1993**] J. Owen, *STEP: An Introduction, Information Geometer*, 47 Stockers Avenue, Winchester SO22 5LB, United Kingdom, 1993.

[**Ouyang 2004**] Ouyang Ying-xiu, Tang Min, Lin Jun-cheng, “Distributed collaborative CAD system based on Web service”, *Journal of Zhejiang University Science*, Vol. 5, No. 5, pp. 579-586, 2004.

- [**Pahng 1998**] Pahng G. D. F., Senin N. and Wallace D., "Distributed modeling and evaluation of product design problems", *Computer Aided Design*, Vol. 30, No. 6, pp. 411-423, 1998.
- [**Papavasileiou 2002**] Papavasileiou. A, Garvros. K, Vasileiadis. V, Savvidis. S, "CAD/CAM interfaces - a review", *Proceedings of 2002 First International IEEE Symposium 'Intelligent Systms'*, Vol. 1, No. 1, pp. 378-382, 2002.
- [**PDMI2 1999**] "PDM status report 09/99", 1999.
- [**Pham 1998**] D. V. Pham and S. S. Dimov, "An Approach to concurrent engineering", *Journal of Engineering Manufacture*, Vol. 212, No. B1, pp. 13-277, 1998.
- [**Qiang 2001**] Qiang L, Zhang Y. F, Nee A. Y. C, "A distributed and collaborative concurrent product design system through the WWW/Internet", *International Journal of Advanced Manufacturing Technology*, Vol. 17, No. 5, pp. 315-322, 2001.
- [**Rafael 2001**] Rafael, B., Eelco Van den Berg, Willem F. Bronsvort, "Collaborative modeling with features", *Proceeding of ASME Design Engineering Technical Conferences*, Vol. 1, pp. 535-546, 2001.
- [**Roller 1991**] D. Roller, "An approach to computer-aided parametric design", *Computer-Aided Design*, Vol. 23, No. 5, pp. 395-391, June 1991.
- [**Roseman 1996**] Roseman M., Greenberg S., "Building real time groupware with GroupKit, a groupware toolit", *Proceeding of ACM transactions on Computer-Humang Interaction*, Vol. 3, No. 1, pp. 66-106, March, 1996.
- [**Rossignac 1990**] Rossignac J. R., "Issues on feature-based editing and interrogation of solid models", *Computer and Graphics*, Vol. 14, No. 2, pp. 149-172, 1990
- [**Shah 1989**] Shah J. J., Rogers M., Sreevalsan P. and Mathew A., "Functional requirements for feature based modeling systems", *CAM-I Report R-89-GM-03*, CAM-I, Texas, USA, 1989.

- [Shah 1995] Shah J. J., Mantyla M., *Parametric and Feature-based CAD/CAM*, New York:Wiley, 1995.
- [Shen 2000] Shen W. M., Maturana F., Norrie D. H., "MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing", *Journal of Intelligent Manufacturing*, Vol. 2, No. 3, pp. 237-251, 2000.
- [Smith 1988] B. Smith, G. Rinaudot, *Product data exchange specification: first working draft*, Gaitherberg, Md. : National Bureau of Standards, US Dept. of Commerce, 1988.
- [Streamline 2006] Autodesk Streamline, Autodesk Inc, www.autodesk.de/streamline (Accessed in December 2006).
- [Sun 2001] Sun J, Zhang YF, Nee AYC, "A distributed multi-agent environment for product design and manufacturing planning", *International Journal of Production Research*, Vol. 39, No. 4, pp. 625-645, 2001.
- [Sung 2001] Sung H. A., Sundararajan V., Smith C., Kannan B., D'Souza R., Sun G. P., Mohole A., Wright P. K., Kim J. H., McMains S., Smith J., Se'quin C. H.. "CyberCut: an internet-based CAD/CAD system", *ASME Journal of Computer Information Science Engineering*, Vol. 1, pp. 52-59, 2001.
- [Tang 2004] Tang S. H., Ma Y. S. and Chen G., "A feature-oriented database framework for web-based CAX applications", *Computer Aided Design and Applications*, ISSN 1686-4360, Vol. 1, No. 1-4, pp. 117-125, 2004.
- [Tay 2003] Tay, Francis EngHock, "CyberCAD: A collaborative approach in 3D-CAD technology in a multimedia-supported environment", *Computer in Industry*, Vol. 52, No. 2, pp. 127-145, October, 2003.
- [Tseng 1994] Y. J. Tseng, S. B. Joshi, "Recognizing multiple interpretation of interacting machining features", *Computer Aided Design*, Vol. 26, No. 9, pp. 667-688, 1994.

- [**Van 2000**] Van den Berg E, Bidarra R, Bronsvoort WF, “Web-based interaction on feature models”, *Proceedings of the Seventh IFIP WG 5.2 Workshop on Geometric Modelling: Fundamentals and Applications*, Parma, Italy, pp. 319–320, 2000.
- [**Vandenbrande 1993**] J. H. Vandenbrande, A. G. Requicha, “Spatial reasoning for the automatic recognition of machinable Feature in solid models”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 12, pp. 1269-1285, 1993.
- [**Windchill 2006**] Windchill, Parametric Technological Company, <http://www.ptc.com/appserver/mkt/products/home.jsp?k=37> (Accessed in December 2006).
- [**Wu 2002**] Wu S. H., Fuh J. Y. H, Nee A. Y. C, “Concurrent process planning and scheduling in distributed virtual manufacturing”, *IIE TRANSACTIONS*, Vol. 34, No. 1, pp. 77-89, 2002.

APPENDIX A

Standard operations transmitted between clients and server for testing part1

```

operation_name: BLOCK
operation_parameter:
  p0=100
  p1=70
  p2=15
location:
  X=0.000000
  Y=0.000000
  Z=0.000000
#
operation_name: Pad
operation_parameter:
  X      : p3=60
  Y      : p4=70
  Z      : p5=15
  Corner_rad : p6=0
  Taper_angle: p7=0
feature_direction
direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=1.000000
X direction
  dir_y[0]=1.000000
  dir_y[1]=0.000000
  dir_y[2]=0.000000
location:
  X=0.000000
  Y=0.000000
  Z=15.000000
#
operation_name: TAPER
operation_parameter:
  taper_angle: p10=10
  taper_direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
netual face:
  face origin:
    x=-0.000000
    y=-0.000000
    z=30.000000
  face direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
  number of taper faces: 4
  taper_face[0]:
    face origin:
      x=50.000013
      y=0.000000
      z=15.000013
    face direction:
      dir_x=0.000000
      dir_y=-1.000000
      dir_z=0.000000
  taper_face[1]:
    face origin:
      x=0.000000
      y=35.000000
      z=15.000000
    face direction:
      dir_x=-1.000000
      dir_y=0.000000
      dir_z=0.000000
  taper_face[2]:
    face origin:
      x=50.000013
      y=70.000000
      z=15.000013
    face direction:
      dir_x=0.000000
      dir_y=1.000000
      dir_z=0.000000
  taper_face[3]:
    face origin:
      x=60.000000
      y=35.000000
      z=22.499988
    face direction:
      dir_x=1.000000
      dir_y=0.000000
      dir_z=0.000000
#
operation_name: TAPER
operation_parameter:
  taper_angle: p11=10
  taper_direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
netual face:
  face origin:
    x=100.000000
    y=-2.644905
    z=15.000000
  face direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
  number of taper faces: 1
  taper_face[0]:
    face origin:
      x=100.000000
      y=35.000000
      z=7.500000
    face direction:
      dir_x=1.000000
      dir_y=0.000000
      dir_z=0.000000
#
operation_name: EXTRUDED
operation_parameter:
  bool_statu=7
  so_creatio=1
  tap_angle=0
  start_dist=0
  end_distan=200
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.990268
  d_y: 0.000000
  d_z: 0.139173
extrude_curves:
  num_curves: 4
  curver_type[0]=3
  curve_start[0]=-15.3620716694672050
  curve_start[1]=64.0431925254759590
  curve_start[2]=71.6678698368494340
  curve_end[0] =-15.3620716694672050
  curve_end[1] =5.9568074745240622
  curve_end[2] =71.6678698368494340
  curver_type[1]=3
  curve_start[0]=-11.6478394800063970
  curve_start[1]=5.9568074745240622
  curve_start[2]=45.2397345740556960

```

```

curve_end[0] =-15.3620716694672050
curve_end[1] =5.9568074745240622
curve_end[2] =71.6678698368494340
curver_type[2]=5
curve_center[0]=-12.6938896147865780
curve_center[1]=35.0000000000000070
curve_center[2]=52.6827680311664110
star_angle =3.3948299839477700
end_angle =6.0299479768215827
radius =29.999999999999110
start_point_x =-11.6478394800065100
start_point_z =64.0431925254759590
start_point_x =45.2397345740563920
end_point_x =-11.6478394800064000
end_point_z =5.9568074745242576
end_point_x =45.2397345740556180
curver_type[3]=3
curve_start[0]=-11.6478394800063980
curve_start[1]=64.0431925254759590
curve_start[2]=45.2397345740557030
curve_end[0] =-15.3620716694672050
curve_end[1] =64.0431925254759590
curve_end[2] =71.6678698368494340
attach face:
face_origin:
point[0]=-12.6938896147865780
point[1]=35.0000000000000070
point[2]=52.6827680311664110
face_dir:
dir_x[0]=0.0000000000000000
dir_x[1]=-1.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=-0.1391731009600654
dir_y[1]=0.0000000000000000
dir_y[2]=0.9902680687415704
dir_z[0]=-0.9902680687415704
dir_z[1]=0.0000000000000000
dir_z[2]=-0.1391731009600654
#
operation_name:HOLLOW
operation_parameter:
type : face
thickness: p18=2
pirerced face: 1
face[0]:
face_point_0[0]=48.677548
face_point_0[1]=35.000000
face_point_0[2]=0.000000
face_dir_0[0] =0.000000
face_dir_0[1] =0.000000
face_dir_0[2] =-1.000000
#
operation_name:BLEND
operation_parameter:
blend_type : 0
b_instance : 0
vrb_tolera : 0.025400
b_sm_overf : 1
b_cl_overf : 1
b_no_overf : 1
num_edgers : 4
radius=5
blend_edge[0]:
edge_type[0]=3001
vertex_count[0]=2
point1[0]=-5.289809
point1[1]=75.289809
point1[2]=0.000000
point2[0]=-0.000000
point2[1]=70.000000
point2[2]=30.000000
point3[0]=-2.644905
point3[1]=72.644905
point3[2]=15.000000
blend_edge[1]:
edge_type[1]=3001
vertex_count[1]=2
point1[0]=-5.289809
point1[1]=75.289809
point1[2]=0.000000
point2[0]=100.000000
point2[1]=-2.644905
point2[2]=15.000000
point3[0]=101.322452
point3[1]=-3.967357
point3[2]=7.500000
blend_edge[2]:
edge_type[2]=3001
vertex_count[2]=2
point1[0]=102.644905
point1[1]=-5.289809
point1[2]=0.000000
point2[0]=100.000000
point2[1]=-2.644905
point2[2]=15.000000
point3[0]=101.322452
point3[1]=-3.967357
point3[2]=7.500000
blend_edge[3]:
edge_type[3]=3001
vertex_count[3]=2
point1[0]=102.644905
point1[1]=75.289809
point1[2]=0.000000
point2[0]=100.000000
point2[1]=72.644905
point2[2]=15.000000
point3[0]=101.322452
point3[1]=73.967357
point3[2]=7.500000
#
operation_name:BLEND
operation_parameter:
blend_type : 0
b_instance : 0
vrb_tolera : 0.025400
b_sm_overf : 1
b_cl_overf : 1
b_no_overf : 1
num_edgers : 14
radius=2
blend_edge[0]:
edge_type[0]=3003
vertex_count[0]=2
point1[0]=0.000000
point1[1]=52.691003
point1[2]=30.000000
point2[0]=0.000000
point2[1]=17.308997
point2[2]=30.000000
point3[0]=-1.053756
point3[1]=35.000000
point3[2]=24.023855
blend_edge[1]:
edge_type[1]=3001
vertex_count[1]=2
point1[0]=0.000000
point1[1]=4.924039
point1[2]=30.000000
point2[0]=0.000000
point2[1]=17.308997
point2[2]=30.000000
point3[0]=0.000000
point3[1]=11.116518
point3[2]=30.000000
blend_edge[2]:
edge_type[2]=3003
vertex_count[2]=2
point1[0]=0.000000
point1[1]=4.924039
point1[2]=30.000000
point2[0]=4.924039

```



```

    point2[1]=0.000000
    point2[2]=30.000000
    point3[0]=1.433333
    point3[1]=1.433333
    point3[2]=30.000000
blend_edge[3]:
    edge_type[3]=3001
    vertex_count[3]=2
    point1[0]=4.924039
    point1[1]=0.000000
    point1[2]=30.000000
    point2[0]=60.000000
    point2[1]=-0.000000
    point2[2]=30.000000
    point3[0]=32.462019
    point3[1]=0.000000
    point3[2]=30.000000
blend_edge[4]:
    edge_type[4]=3001
    vertex_count[4]=2
    point1[0]=62.644905
    point1[1]=-2.644905
    point1[2]=15.000000
    point2[0]=60.000000
    point2[1]=-0.000000
    point2[2]=30.000000
    point3[0]=61.322452
    point3[1]=-1.322452
    point3[2]=22.500000
blend_edge[5]:
    edge_type[5]=3001
    vertex_count[5]=2
    point1[0]=95.075961
    point1[1]=-2.644905
    point1[2]=15.000000
    point2[0]=62.644905
    point2[1]=-2.644905
    point2[2]=15.000000
    point3[0]=78.860433
    point3[1]=-2.644905
    point3[2]=15.000000
blend_edge[6]:
    edge_type[6]=3003
    vertex_count[6]=2
    point1[0]=95.075961
    point1[1]=-2.644905
    point1[2]=15.000000
    point2[0]=100.000000
    point2[1]=2.279134
    point2[2]=15.000000
    point3[0]=98.566667
    point3[1]=-1.211572
    point3[2]=15.000000
blend_edge[7]:
    edge_type[7]=3001
    vertex_count[7]=2
    point1[0]=100.000000
    point1[1]=67.720866
    point1[2]=15.000000
    point2[0]=100.000000
    point2[1]=2.279134
    point2[2]=15.000000
    point3[0]=100.000000
    point3[1]=35.000000
    point3[2]=15.000000
blend_edge[8]:
    edge_type[8]=3003
    vertex_count[8]=2
    point1[0]=100.000000
    point1[1]=67.720866
    point1[2]=15.000000
    point2[0]=95.075961
    point2[1]=72.644905
    point2[2]=15.000000
    point3[0]=98.566667
    point3[1]=71.211572
    point3[2]=15.000000
blend_edge[9]:
    edge_type[9]=3001
    vertex_count[9]=2
    point1[0]=62.644905
    point1[1]=72.644905
    point1[2]=15.000000
    point2[0]=95.075961
    point2[1]=72.644905
    point2[2]=15.000000
    point3[0]=78.860433
    point3[1]=72.644905
    point3[2]=15.000000
blend_edge[10]:
    edge_type[10]=3001
    vertex_count[10]=2
    point1[0]=62.644905
    point1[1]=72.644905
    point1[2]=15.000000
    point2[0]=60.000000
    point2[1]=70.000000
    point2[2]=30.000000
    point3[0]=61.322452
    point3[1]=71.322452
    point3[2]=22.500000
blend_edge[11]:
    edge_type[11]=3001
    vertex_count[11]=2
    point1[0]=4.924039
    point1[1]=70.000000
    point1[2]=30.000000
    point2[0]=60.000000
    point2[1]=70.000000
    point2[2]=30.000000
    point3[0]=32.462019
    point3[1]=70.000000
    point3[2]=30.000000
blend_edge[12]:
    edge_type[12]=3003
    vertex_count[12]=2
    point1[0]=4.924039
    point1[1]=70.000000
    point1[2]=30.000000
    point2[0]=0.000000
    point2[1]=65.075961
    point2[2]=30.000000
    point3[0]=1.433333
    point3[1]=68.566667
    point3[2]=30.000000
blend_edge[13]:
    edge_type[13]=3001
    vertex_count[13]=2
    point1[0]=0.000000
    point1[1]=52.691003
    point1[2]=30.000000
    point2[0]=0.000000
    point2[1]=65.075961
    point2[2]=30.000000
    point3[0]=0.000000
    point3[1]=58.883482
    point3[2]=30.000000
#
operation_name:BLEND
operation_parameter:
    blend_type : 0
    b_instance : 0
    vrb_tolera : 0.025400
    b_sm_overf : 1
    b_cl_overf : 1
    b_no_overf : 1
    num_edgers : 10
    radius=3
blend_edge[0]:
    edge_type[0]=3003
    vertex_count[0]=2
    point1[0]=58.321801

```

```

point1[1]=-0.291416
point1[2]=28.347296
point2[0]=59.978871
point2[1]=1.678199
point2[2]=30.000000
point3[0]=59.403377
point3[1]=0.392624
point3[2]=29.532089
blend_edge[1]:
edge_type[1]=3003
vertex_count[1]=2
point1[0]=59.978871
point1[1]=1.678199
point1[2]=30.000000
point2[0]=60.000000
point2[1]=1.969616
point2[2]=30.000000
point3[0]=59.994711
point3[1]=1.823367
point3[2]=30.000000
blend_edge[2]:
edge_type[2]=3001
vertex_count[2]=2
point1[0]=60.000000
point1[1]=1.969616
point1[2]=30.000000
point2[0]=60.000000
point2[1]=68.030384
point2[2]=30.000000
point3[0]=60.000000
point3[1]=35.000000
point3[2]=30.000000
blend_edge[3]:
edge_type[3]=3003
vertex_count[3]=2
point1[0]=60.000000
point1[1]=68.030384
point1[2]=30.000000
point2[0]=59.978871
point2[1]=68.321801
point2[2]=30.000000
point3[0]=59.994711
point3[1]=68.176633
point3[2]=30.000000
blend_edge[4]:
edge_type[4]=3003
vertex_count[4]=2
point1[0]=59.978871
point1[1]=68.321801
point1[2]=30.000000
point2[0]=58.321801
point2[1]=70.291416
point2[2]=28.347296
point3[0]=59.403377
point3[1]=69.607376
point3[2]=29.532089
blend_edge[5]:
edge_type[5]=3003
vertex_count[5]=2
point1[0]=60.966705
point1[1]=-2.936321
point1[2]=13.347296
point2[0]=62.623776
point2[1]=-0.966705
point2[2]=15.000000
point3[0]=62.048281
point3[1]=-2.252281
point3[2]=14.532089
blend_edge[6]:
edge_type[6]=3003
vertex_count[6]=2
point1[0]=62.644905
point1[1]=-0.675289
point1[2]=15.000000
point2[0]=62.623776
point2[1]=-0.966705
point2[2]=15.000000
point3[0]=62.639616
point3[1]=-0.821538
point3[2]=15.000000
blend_edge[7]:
edge_type[7]=3001
vertex_count[7]=2
point1[0]=62.644905
point1[1]=-0.675289
point1[2]=15.000000
point2[0]=62.644905
point2[1]=70.675289
point2[2]=15.000000
point3[0]=62.644905
point3[1]=35.000000
point3[2]=15.000000
blend_edge[8]:
edge_type[8]=3003
vertex_count[8]=2
point1[0]=62.623776
point1[1]=70.966705
point1[2]=15.000000
point2[0]=62.644905
point2[1]=70.675289
point2[2]=15.000000
point3[0]=62.639616
point3[1]=70.821538
point3[2]=15.000000
blend_edge[9]:
edge_type[9]=3003
vertex_count[9]=2
point1[0]=62.623776
point1[1]=70.966705
point1[2]=15.000000
point2[0]=60.966705
point2[1]=72.936321
point2[2]=13.347296
point3[0]=62.048281
point3[1]=72.252281
point3[2]=14.532089
#
operation_name:BLEND
operation_parameter:
blend_type : 0
b_instance : 0
vrbl_tolera : 0.025400
b_sm_overf : 1
b_cl_overf : 1
b_no_overf : 1
num_edges : 4
radius=3
blend_edge[0]:
edge_type[0]=3001
vertex_count[0]=2
point1[0]=-3.258956
point1[1]=73.258956
point1[2]=0.000000
point2[0]=1.678199
point2[1]=68.321801
point2[2]=28.000000
point3[0]=-0.790378
point3[1]=70.790378
point3[2]=14.000000
blend_edge[1]:
edge_type[1]=3001
vertex_count[1]=2
point1[0]=-3.258956
point1[1]=-3.258956
point1[2]=0.000000
point2[0]=1.678199
point2[1]=1.678199
point2[2]=28.000000
point3[0]=-0.790378
point3[1]=-0.790378
point3[2]=14.000000
blend_edge[2]:

```



```

edge_type[2]=3001
vertex_count[2]=2
point1[0]=100.614051
point1[1]=-3.258956
point1[2]=0.000000
point2[0]=98.321801
point2[1]=-0.966705
point2[2]=13.000000
point3[0]=99.467926
point3[1]=-2.112831
point3[2]=6.500000
blend_edge[3]:
edge_type[3]=3001
vertex_count[3]=2
point1[0]=100.614051
point1[1]=73.258956
point1[2]=0.000000
point2[0]=98.321801
point2[1]=70.966705
point2[2]=13.000000
point3[0]=99.467926
point3[1]=72.112831
point3[2]=6.500000
#
operation_name:EXTRUDED
operation_parameter:
  bool_statu=6
  so_creation=1
  tap_angle=0
  start_dist=0
  end_distan=18
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.000000
  d_y: 0.000000
  d_z: -1.000000
extrude_curves:
  num_curves: 4
  curver_type[0]=3
  curve_start[0]=35.6781992623545680
  curve_start[1]=68.3218007376454470
  curve_start[2]=27.9999999999999960
  curve_end[0]=35.6781992623545530
  curve_end[1]=1.6781992623545605
  curve_end[2]=27.9999999999999960
  curver_type[1]=3
  curve_start[0]=35.6781992623545530
  curve_start[1]=1.6781992623545605
  curve_start[2]=27.9999999999999960
  curve_end[0]=37.6781992623545530
  curve_end[1]=1.6781992623545605
  curve_end[2]=27.9999999999999960
  curver_type[2]=3
  curve_start[0]=37.6781992623545530
  curve_start[1]=1.6781992623545605
  curve_start[2]=27.9999999999999960
  curve_end[0]=37.6781992623545610
  curve_end[1]=68.3218007376454470
  curve_end[2]=27.9999999999999960
  curver_type[3]=3
  curve_start[0]=37.6781992623545610
  curve_start[1]=68.3218007376454470
  curve_start[2]=27.9999999999999960
  curve_end[0]=35.6781992623545680
  curve_end[1]=68.3218007376454470
  curve_end[2]=27.9999999999999960
attach_face:
face_origin:
  point[0]=37.6781992623545610
  point[1]=68.3218007376454470
  point[2]=27.9999999999999960
face_dir:
  dir_x[0]=-1.0000000000000000
  dir_x[1]=0.0000000000000000
  dir_x[2]=0.0000000000000000

```

```

dir_y[0]=0.0000000000000000
dir_y[1]=1.0000000000000000
dir_y[2]=0.0000000000000000
dir_z[0]=0.0000000000000000
dir_z[1]=0.0000000000000000
dir_z[2]=-1.0000000000000000
#
operation_name:MIRROR
parent feature:
  num of parent feature=1
  INSTANCE[0](17)/EXTRUDED(17)
  Boss
  parent_direction:
    X: 0.000000
    Y: 0.000000
    Z: -1.000000
  parent_location:
    X: 35.678199
    Y: 68.321801
    Z: 28.000000
mirror_plane:
  plane_origin:
    o_x=36.678199
    o_y=7.350977
    o_z=19.000000
  plane_norm:
    dir_x[0]=1.000000
    dir_x[1]=0.000000
    dir_x[2]=0.000000
#
operation_name:EXTRUDED
operation_parameter:
  bool_statu=7
  so_creation=1
  tap_angle=0
  start_dist=0
  end_distan=200
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.990268
  d_y: 0.000000
  d_z: 0.139173
extrude_curves:
  num_curves: 4
  curver_type[0]=3
  curve_start[0]=-11.6478394800063980
  curve_start[1]=64.0431925254759590
  curve_start[2]=45.2397345740557030
  curve_end[0]=-15.3620716694672050
  curve_end[1]=64.0431925254759590
  curve_end[2]=71.6678698368494340
  curver_type[1]=5
  curve_center[0]=-12.6938896147865780
  curve_center[1]=35.0000000000000070
  curve_center[2]=52.6827680311664110
  star_angle=3.3948299839477700
  end_angle=6.0299479768215827
  radius=29.9999999999999110
  start_point_x=-11.6478394800065100
  start_point_z=64.0431925254759590
  start_point_x=45.2397345740563920
  end_point_x=-11.6478394800064000
  end_point_z=5.9568074745242576
  end_point_x=45.2397345740556180
  curver_type[2]=3
  curve_start[0]=-11.6478394800063970
  curve_start[1]=5.9568074745240622
  curve_start[2]=45.2397345740556960
  curve_end[0]=-15.3620716694672050
  curve_end[1]=5.9568074745240622
  curve_end[2]=71.6678698368494340
  curver_type[3]=3
  curve_start[0]=-15.3620716694672050
  curve_start[1]=64.0431925254759590
  curve_start[2]=71.6678698368494340

```



```

curve_end[0] =-15.3620716694672050
curve_end[1] =5.9568074745240622
curve_end[2] =71.6678698368494340
attach face:
face_origin:
point[0]=-12.6938896147865780
point[1]=35.0000000000000070
point[2]=52.6827680311664110
face_dir:
dir_x[0]=0.0000000000000000
dir_x[1]=-1.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=-0.1391731009600654
dir_y[1]=0.0000000000000000
dir_y[2]=0.9902680687415704
dir_z[0]=-0.9902680687415704
dir_z[1]=0.0000000000000000
dir_z[2]=-0.1391731009600654
#
operation_name:EXTRUDED
operation_parameter:
bool_statu=6
so_creatio=1
tap_angle=0
start_dist=0
end_distan=18
first_offs=0.000000
secon_offs=0.000000
direction:
d_x: 0.000000
d_y: 0.000000
d_z: -1.000000
extrude_curves:
num_curves: 1
curver_type[0]=5
curve_center[0]=36.6781992623545610
curve_center[1]=19.6781992623545610
curve_center[2]=27.9999999999999960
star_angle =0.0000000000000000
end_angle =6.2831853071795862
radius =3.0000000000000000
start_point_x =39.6781992623545610
start_point_z =19.6781992623545610
start_point_x =27.9999999999999960
end_point_x =39.6781992623545610
end_point_z =19.6781992623545610
end_point_x =27.9999999999999960
attach face:
face_origin:
point[0]=36.6781992623545610
point[1]=19.6781992623545610
point[2]=27.9999999999999960
face_dir:
dir_x[0]=1.0000000000000000
dir_x[1]=0.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=0.0000000000000000
dir_y[1]=-1.0000000000000000
dir_y[2]=0.0000000000000000
dir_z[0]=0.0000000000000000
dir_z[1]=0.0000000000000000
dir_z[2]=-1.0000000000000000
#
operation_name:RECTANGULAR_ARRAY
parent feature:
num of parent feature=1
INSTANCE[0](21)/EXTRUDED(21)
Boss
parent_direction:
X: 0.000000
Y: 0.000000
Z: -1.000000
parent_location:
X: 36.678199
Y: 19.678199
Z: 28.000000

```

```

array csys:
csys_origin:
o_x=36.678199
o_y=19.678199
o_z=28.000000
csys_matrix:
dir_x[0]=1.000000
dir_x[1]=0.000000
dir_x[2]=0.000000
dir_y[0]=0.000000
dir_y[1]=1.000000
dir_y[2]=0.000000
dir_z[0]=0.000000
dir_z[1]=0.000000
dir_z[2]=1.000000
array param:
num_x: p47=2
num_y: p49=2
x_offset: p48=-12
y_offset: p50=30
#
operation_name:EXTRUDED
operation_parameter:
bool_statu=7
so_creatio=1
tap_angle=0
start_dist=0
end_distan=18
first_offs=0.000000
secon_offs=0.000000
direction:
d_x: 0.000000
d_y: 0.000000
d_z: -1.000000
extrude_curves:
num_curves: 1
curver_type[0]=5
curve_center[0]=36.6781992623545610
curve_center[1]=19.6781992623545610
curve_center[2]=27.9999999999999960
star_angle =0.0000000000000000
end_angle =6.2831853071795862
radius =1.5000000000000000
start_point_x =38.1781992623545610
start_point_z =19.6781992623545610
start_point_x =27.9999999999999960
end_point_x =38.1781992623545610
end_point_z =19.6781992623545610
end_point_x =27.9999999999999960
attach face:
face_origin:
point[0]=36.6781992623545610
point[1]=19.6781992623545610
point[2]=27.9999999999999960
face_dir:
dir_x[0]=1.0000000000000000
dir_x[1]=0.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=0.0000000000000000
dir_y[1]=-1.0000000000000000
dir_y[2]=0.0000000000000000
dir_z[0]=0.0000000000000000
dir_z[1]=0.0000000000000000
dir_z[2]=-1.0000000000000000
#
operation_name:RECTANGULAR_ARRAY
parent feature:
num of parent feature=1
INSTANCE[0](24)/EXTRUDED
Cut
parent_direction:
X: 0.000000
Y: 0.000000
Z: -1.000000
parent_location:
X: 36.678199

```

```

        Y: 19.678199
        Z: 28.000000
array csys:
  csys_origin:
    o_x=36.678199
    o_y=19.678199
    o_z=28.000000
  csys_matrix:
    dir_x[0]=1.000000
    dir_x[1]=0.000000
    dir_x[2]=0.000000
    dir_y[0]=0.000000
    dir_y[1]=1.000000
    dir_y[2]=0.000000
    dir_z[0]=0.000000
    dir_z[1]=0.000000
    dir_z[2]=1.000000
array param:
  num_x: p55=2
  num_y: p57=2
  x_offset: p56=-12
  y_offset: p58=30
#
operation_name: BOSS(26)
operation_parm:
  diameter : p59=12
  height : p60=6
  taper_angle: p61=0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=-1.000000
location:
  X=83.321801
  Y=14.033295
  Z=13.000000
#
operation_name:MIRROR
parent feature:
  num of parent feature=1
  INSTANCE[0](26)/BOSS
  Boss
  parent_direction:
    X: 0.000000
    Y: 0.000000
    Z: -1.000000
  parent_location:
    X: 83.321801
    Y: 14.033295
    Z: 13.000000
mirror_plane:
  plane_origin:
    o_x= 83.321801
    o_y= 34.033295
    o_z= 13.000000
  plane_norm:
    dir_x[0]=0.000000
    dir_x[1]=1.000000
    dir_x[2]=0.000000
#
operation_name: BOSS
operation_parm:
  diameter : p71=8
  height : p72=6
  taper_angle: p73=0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=-1.000000
location:
  X=83.321801
  Y=14.033295
  Z=7.000000
#
operation_name:MIRROR
parent feature:

```

```

  num of parent feature=1
  INSTANCE[0](28)/BOSS(28)
  Boss
  parent_direction:
    X: 0.000000
    Y: 0.000000
    Z: -1.000000
  parent_location:
    X: 83.321801
    Y: 14.033295
    Z: 7.000000
mirror_plane:
  plane_origin:
    o_x= 83.321801
    o_y= 34.033295
    o_z= 13.000000
  plane_norm:
    dir_x[0]=0.000000
    dir_x[1]=1.000000
    dir_x[2]=0.000000
#
operation_name:SIMPLE_HOLE
operation_parm:
  diameter : p83=10
  depth : p84=5
  Taper_angle: p85=0
  through_flag:0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=-1.000000
location:
  X=83.321801
  Y=14.033295
  Z=15.000000
#
operation_name:MIRROR
parent feature:
  num of parent feature=1
  INSTANCE[0](30)/SIMPLE_HOLE(30)
  Cut
  parent_direction:
    X: 0.000000
    Y: 0.000000
    Z: -1.000000
  parent_location:
    X: 83.321801
    Y: 14.033295
    Z: 15.000000
mirror_plane:
  plane_origin:
    o_x= 83.321801
    o_y= 34.033295
    o_z= 13.000000
  plane_norm:
    dir_x[0]=0.000000
    dir_x[1]=1.000000
    dir_x[2]=0.000000
#
operation_name:SIMPLE_HOLE
operation_parm:
  diameter : p94=4
  depth : p95=50
  Taper_angle: p96=0
  through_flag:0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=-1.000000
location:
  X=83.321801
  Y=14.033295
  Z=10.000000
#
operation_name:MIRROR
parent feature:

```



```

num of parent feature=1
  INSTANCE[0](32)/SIMPLE_HOLE
  Cut
  parent_direction:
    X: 0.000000
    Y: 0.000000
    Z: -1.000000
  parent_location:
    X: 83.321801
    Y: 14.033295
    Z: 10.000000
mirror_plane:
  plane_origin:
    o_x= 83.321801
    o_y= 34.033295
    o_z= 13.000000
  plane_norm:
    dir_x[0]=0.000000
    dir_x[1]=1.000000
    dir_x[2]=0.000000
#
operation_name: BOSS
operation_parm:
  diameter : p105=6
  height : p106=12
  taper_angle: p107=0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=-1.000000
location:
  X=73.321801
  Y=35.000000
  Z=13.000000
#
operation_name:SIMPLE_HOLE
opeation_parm:
  diameter : p114=3
  depth : p115=50
  Taper_angle :p116=0
  through_flag:0
feature_direction
  dir_x[0]=0.000000
  dir_x[1]=0.000000
  dir_x[2]=1.000000
location:
  X=73.321801
  Y=35.000000
  Z=1.000000
#
operation_name:EXTRUDED
operation_parameter:
  bool_statu=6
  so_creatio=1
  tap_angle=0
  start_dist=0
  end_distan=2
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.000000
  d_y: -1.000000
  d_z: 0.000000
extrude_curves:
  num_curves: 3
  curver_type[0]=3
  curve_start[0]=81.8218007376454470
  curve_start[1]=36.0000000000000070
  curve_start[2]=13.0000000000000020
  curve_end[0] =74.8218007376454470
  curve_end[1] =36.0000000000000070
  curve_end[2] =2.0000000000000018
  curver_type[1]=3
  curve_start[0]=74.8218007376454470
  curve_start[1]=36.0000000000000070
  curve_start[2]=2.0000000000000018

```

```

  curve_end[0] =74.8218007376454470
  curve_end[1] =36.0000000000000070
  curve_end[2] =13.0000000000000020
  curver_type[2]=3
  curve_start[0]=74.8218007376454470
  curve_start[1]=36.0000000000000070
  curve_start[2]=13.0000000000000020
  curve_end[0] =81.8218007376454470
  curve_end[1] =36.0000000000000070
  curve_end[2] =13.0000000000000020
attach face:
  face_origin:
    point[0]=74.8218007376454470
    point[1]=36.0000000000000070
    point[2]=13.0000000000000020
  face_dir:
    dir_x[0]=0.0000000000000000
    dir_x[1]=0.0000000000000000
    dir_x[2]=1.0000000000000000
    dir_y[0]=-1.0000000000000000
    dir_y[1]=0.0000000000000000
    dir_y[2]=0.0000000000000000
    dir_z[0]=0.0000000000000000
    dir_z[1]=-1.0000000000000000
    dir_z[2]=0.0000000000000000
#
operation_name:CIRCULAR_ARRAY
parent feature:
  num of parent feature=1
  INSTANCE[0](38)/EXTRUDED
  Boss
  parent_direction:
    X: 0.000000
    Y: -1.000000
    Z: 0.000000
  parent_location:
    X: 81.821801
    Y: 36.000000
    Z: 13.000000
array csys:
  array_origin:
    o_x=73.321801
    o_y=35.000000
    o_z=13.000000
  array_axis:
    dir_x[0]=0.000000
    dir_x[1]=0.000000
    dir_x[2]=-1.000000
array param:
  number: p128=4
  angle: p129=90
#
operation_name:BLEND
operation_parameter:
  blend_type : 0
  b_instance : 0
  vrb_tolera : 0.025400
  b_sm_overf : 1
  b_cl_overf : 1
  b_no_overf : 1
  num_edges : 2
  radius=3
blend_edge[0]:
  edge_type[0]=3001
  vertex_count[0]=2
  point1[0]=60.966705
  point1[1]=-0.966705
  point1[2]=13.000000
  point2[0]=60.966705
  point2[1]=70.966705
  point2[2]=13.000000
  point3[0]=60.966705
  point3[1]=35.000000
  point3[2]=13.000000
blend_edge[1]:
  edge_type[1]=3001

```



```

vertex_count[1]=2
point1[0]=58.321801
point1[1]=1.678199
point1[2]=28.000000
point2[0]=58.321801
point2[1]=68.321801
point2[2]=28.000000
point3[0]=58.321801
point3[1]=35.000000
point3[2]=28.000000
#
operation_name:EXTRUDED
operation_parameter:
  bool_statu=7
  so_creatio=1
  tap_angle=0
  start_dist=0
  end_distan=10
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 1.000000
  d_y: 0.000000
  d_z: 0.000000
extrude_curves:
  num_curves: 4
  curver_type[0]=3
  curve_start[0]=57.4827011064681490
  curve_start[1]=16.6781992623545640
  curve_start[2]=21.4790554669992130
  curve_end[0]=57.4827011064681490
  curve_end[1]=16.6781992623545750
  curve_end[2]=94.5333662201325070
  curver_type[1]=5
  curve_center[0]=57.4827011064681490
  curve_center[1]=19.6781992623545640
  curve_center[2]=21.4790554669992130
  star_angle=0.0000000000000000
  end_angle=3.1415926535897931
  radius=3.0000000000000000
  start_point_x=57.4827011064681490
  start_point_z=22.6781992623545640
  start_point_x=21.4790554669992130
  end_point_x=57.4827011064681490
  end_point_z=16.6781992623545640
  end_point_x=21.4790554669992130
  curver_type[2]=3
  curve_start[0]=57.4827011064681490
  curve_start[1]=22.6781992623545640
  curve_start[2]=21.4790554669992130
  curve_end[0]=57.4827011064681490
  curve_end[1]=22.6781992623545710
  curve_end[2]=94.5333662201325070
  curver_type[3]=3
  curve_start[0]=57.4827011064681490
  curve_start[1]=16.6781992623545750
  curve_start[2]=94.5333662201325070
  curve_end[0]=57.4827011064681490
  curve_end[1]=22.6781992623545710
  curve_end[2]=94.5333662201325070
attach_face:
  face_origin:
    point[0]=57.4827011064681490
    point[1]=19.6781992623545640
    point[2]=21.4790554669992130
  face_dir:
    dir_x[0]=0.0000000000000000
    dir_x[1]=1.0000000000000000
    dir_x[2]=0.0000000000000000
    dir_y[0]=0.0000000000000000
    dir_y[1]=0.0000000000000000
    dir_y[2]=-1.0000000000000000
    dir_z[0]=-1.0000000000000000
    dir_z[1]=0.0000000000000000
    dir_z[2]=0.0000000000000000
#
operation_name:TAPER
operation_parameter:
  taper_angle:p144=12
  taper_direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
netual face:
  face origin:
    x=25.678199
    y=68.321801
    z=28.000000
  face direction:
    dir_x=0.000000
    dir_y=0.000000
    dir_z=1.000000
  number of taper faces: 2
  taper_face[0]:
    face origin:
      x=24.678199
      y=68.321801
      z=19.000000
    face direction:
      dir_x=0.000000
      dir_y=1.000000
      dir_z=0.000000
  taper_face[1]:
    face origin:
      x=24.678199
      y=1.678199
      z=19.000000
    face direction:

```

```

    dir_x=0.000000
    dir_y=-1.000000
    dir_z=0.000000
#
operation_name:TAPER
operation_parameter:
    taper_angle: p145=12
    taper_direction:
        dir_x=0.000000
        dir_y=0.000000
        dir_z=1.000000
netual face:
    face origin:
        x=37.678199
        y=68.321801
        z=28.000000
    face direction:
        dir_x=0.000000
        dir_y=0.000000
        dir_z=1.000000
    number of taper faces: 2
taper_face[0]:
    face origin:
        x=36.678199
        y=68.321801
        z=19.000000
    face direction:
        dir_x=0.000000
        dir_y=1.000000
        dir_z=0.000000
taper_face[1]:
    face origin:
        x=36.678199
        y=1.678199
        z=19.000000
    face direction:
        dir_x=0.000000
        dir_y=-1.000000
        dir_z=0.000000
#
```

APPENDIX B

Standard operations transmitted between clients and server for testing part 2

```
operation_name:REVOLVED
operation_parameter:
  bool_statu=4
  so_creatio=1
  start_angle=0.000000
  end_angle=360.000000
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.000000
  d_y: 0.000000
  d_z: 1.000000
axis_point:
  x: 0.0000000000000000
  y: 0.0000000000000000
  z: -40.0000000000000000
revolve_curves:
  num_curves: 6
  curver_type[0]=3
  curve_start[0]=0.0000000000000000
  curve_start[1]=0.0000000000000000
  curve_start[2]=-29.9999999999999720
  curve_end[0] =30.0000000000000000
  curve_end[1] =0.0000000000000000
  curve_end[2] =-29.9999999999999610
  curver_type[1]=3
  curve_start[0]=0.0000000000000000
  curve_start[1]=0.0000000000000000
  curve_start[2]=70.0000000000000280
  curve_end[0] =0.0000000000000000
  curve_end[1] =0.0000000000000000
  curve_end[2] =-29.9999999999999720
  curver_type[2]=3
  curve_start[0]=15.0000000000000000
  curve_start[1]=0.0000000000000000
  curve_start[2]=70.0000000000000280
  curve_end[0] =0.0000000000000000
  curve_end[1] =0.0000000000000000
  curve_end[2] =70.0000000000000280
  curver_type[3]=3
  curve_start[0]=28.4697350788555960
  curve_start[1]=0.0000000000000000
  curve_start[2]=29.4590794763433620
  curve_end[0] =15.0000000000000000
  curve_end[1] =0.0000000000000000
  curve_end[2] =70.0000000000000280
  curver_type[4]=5
  curve_center[0]=0.0000000000000052
  curve_center[1]=0.0000000000000000
  curve_center[2]=20.000000000000390
  star_angle =-0.3207755539442190
  end_angle =0.0000000000000001
  radius =30.0000000000000000
  start_point_x =28.4697350788555920
  start_point_z =0.0000000000000000
  start_point_x =29.4590794763433690
  end_point_x =30.0000000000000040
  end_point_z =0.0000000000000000
  end_point_x =20.000000000000360
  curver_type[5]=3
  curve_start[0]=30.0000000000000000
  curve_start[1]=0.0000000000000000
  curve_start[2]=-29.9999999999999610
  curve_end[0] =30.0000000000000070
  curve_end[1] =0.0000000000000000
  curve_end[2] =20.000000000000360
attach_face:
face_origin:
point[0]=0.0000000000000052
```

```
point[1]=0.0000000000000000
point[2]=20.000000000000390
face_dir:
  dir_x[0]=1.0000000000000000
  dir_x[1]=0.0000000000000000
  dir_x[2]=0.0000000000000000
  dir_y[0]=0.0000000000000000
  dir_y[1]=0.0000000000000000
  dir_y[2]=-1.0000000000000000
  dir_z[0]=0.0000000000000000
  dir_z[1]=1.0000000000000000
  dir_z[2]=0.0000000000000000
#
operation_name:HOLLOW
operation_parameter:
  type :face
  thickness: p2= 3.000000
  pircerd face: 2
face[0]:
  face_point_0[0]=0.000000
  face_point_0[1]=0.000000
  face_point_0[2]=70.000000
  face_dir_0[0] =0.000000
  face_dir_0[1] =0.000000
  face_dir_0[2] =1.000000
face[1]:
  face_point_1[0]=0.000000
  face_point_1[1]=0.000000
  face_point_1[2]=-30.000000
  face_dir_1[0] =0.000000
  face_dir_1[1] =0.000000
  face_dir_1[2] =-1.000000
#
operation_name:EXTRUDED
operation_parameter:
  bool_statu=7
  so_creatio=1
  tap_angle=0.000000
  start_dist=0.000000
  end_distan=18.000000
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.000000
  d_y: -1.000000
  d_z: 0.000000
extrude_curves:
  num_curves: 4
  curver_type[0]=5
  curve_center[0]=-5.0000000000000044
  curve_center[1]=30.0000000000000000
  curve_center[2]=7.5000000000000311
  star_angle =1.5707963267948966
  end_angle =4.7123889803846906
  radius =2.5000000000000000
  start_point_x =-5.0000000000000044
  start_point_z =30.0000000000000000
  start_point_x =5.0000000000000311
  end_point_x =-5.0000000000000027
  end_point_z =30.0000000000000000
  end_point_x =10.0000000000000320
  curver_type[1]=3
  curve_start[0]=-5.0000000000000053
  curve_start[1]=30.0000000000000000
  curve_start[2]=5.0000000000000302
  curve_end[0] =4.9999999999999929
  curve_end[1] =30.0000000000000000
  curve_end[2] =5.0000000000000284
  curver_type[2]=5
```



```

curve_center[0]=4.99999999999956
curve_center[1]=30.00000000000000
curve_center[2]=7.5000000000000311
star_angle =1.5707963267948970
end_angle =1.5707963267948970
radius =2.5000000000000000
start_point_x =4.999999999999947
start_point_z =30.0000000000000000
start_point_x =10.0000000000000320
end_point_x =4.999999999999947
end_point_z =30.0000000000000000
end_point_x =5.0000000000000311
curver_type[3]=3
curve_start[0]=4.999999999999938
curve_start[1]=30.0000000000000000
curve_start[2]=10.0000000000000300
curve_end[0] =5.0000000000000044
curve_end[1] =30.0000000000000000
curve_end[2] =10.0000000000000300
attach face:
face_origin:
point[0]=4.999999999999956
point[1]=30.0000000000000000
point[2]=7.5000000000000311
face_dir:
dir_x[0]=1.0000000000000000
dir_x[1]=0.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=0.0000000000000000
dir_y[1]=0.0000000000000000
dir_y[2]=-1.0000000000000000
dir_z[0]=0.0000000000000000
dir_z[1]=1.0000000000000000
dir_z[2]=0.0000000000000000
#
operation_name:EXTRUDED
operation_parameter:
bool_statu=7
so_creatio=1
tap_angle=0.000000
start_dist=0.000000
end_distan=18.000000
first_offs=0.000000
secon_offs=0.000000
direction:
d_x: 0.000000
d_y: -1.000000
d_z: 0.000000
extrude_curves:
num_curves: 4
curver_type[0]=5
curve_center[0]=5.000000000000027
curve_center[1]=30.0000000000000000
curve_center[2]=-2.499999999999685
star_angle =1.5707963267948966
end_angle =4.7123889803846906
radius =2.5000000000000000
start_point_x =5.000000000000027
start_point_z =30.0000000000000000
start_point_x =4.999999999999680
end_point_x =5.0000000000000009
end_point_z =30.0000000000000000
end_point_x =0.0000000000000315
curver_type[1]=3
curve_start[0]=5.000000000000027
curve_start[1]=30.0000000000000000
curve_start[2]=-4.999999999999689
curve_end[0] =4.999999999999893
curve_end[1] =30.0000000000000000
curve_end[2] =-4.999999999999707
curver_type[2]=5
curve_center[0]=4.999999999999893
curve_center[1]=30.0000000000000000
curve_center[2]=-2.499999999999685
star_angle =1.5707963267948970
end_angle =1.5707963267948970
radius =2.5000000000000000
start_point_x =4.999999999999885
start_point_z =30.0000000000000000
start_point_x =0.0000000000000315
end_point_x =4.999999999999885
end_point_z =30.0000000000000000
end_point_x =-4.999999999999680
curver_type[3]=3
curve_start[0]=4.999999999999893
curve_start[1]=30.0000000000000000
curve_start[2]=0.0000000000000304
curve_end[0] =-5.0000000000000018
curve_end[1] =30.0000000000000000
curve_end[2] =0.0000000000000321
attach face:
face_origin:
point[0]=4.999999999999893
point[1]=30.0000000000000000
point[2]=-2.499999999999685
face_dir:
dir_x[0]=1.0000000000000000
dir_x[1]=0.0000000000000000
dir_x[2]=0.0000000000000000
dir_y[0]=0.0000000000000000
dir_y[1]=0.0000000000000000
dir_y[2]=-1.0000000000000000
dir_z[0]=0.0000000000000000
dir_z[1]=1.0000000000000000
dir_z[2]=0.0000000000000000
#
operation_name:EXTRUDED
operation_parameter:
bool_statu=7
so_creatio=1
tap_angle=0.000000
start_dist=0.000000
end_distan=18.000000
first_offs=0.000000
secon_offs=0.000000
direction:
d_x: 0.000000
d_y: -1.000000
d_z: 0.000000
extrude_curves:
num_curves: 4
curver_type[0]=5
curve_center[0]=4.999999999999964
curve_center[1]=30.0000000000000000
curve_center[2]=-12.499999999999570
star_angle =1.5707963267948966
end_angle =4.7123889803846906
radius =2.5000000000000000
start_point_x =4.999999999999964
start_point_z =30.0000000000000000
start_point_x =-14.999999999999570
end_point_x =4.999999999999947
end_point_z =30.0000000000000000
end_point_x =9.999999999999574
curver_type[1]=3
curve_start[0]=4.999999999999964
curve_start[1]=30.0000000000000000
curve_start[2]=-14.999999999999590
curve_end[0] =4.999999999999813
curve_end[1] =30.0000000000000000
curve_end[2] =-14.999999999999610
curver_type[2]=5
curve_center[0]=5.0000000000000036
curve_center[1]=30.0000000000000000
curve_center[2]=-12.499999999999610
star_angle =1.5707963267949050
end_angle =1.5707963267949057
radius =2.5000000000000000
start_point_x =4.999999999999822
start_point_z =30.0000000000000000
start_point_x =9.999999999999609
end_point_x =4.999999999999805

```

```

end_point_z =30.000000000000000
end_point_x =-14.999999999999610
curver_type[3]=3
curve_start[0]=-4.99999999999964
curve_start[1]=30.000000000000000
curve_start[2]=-9.999999999999574
curve_end[0] =4.999999999999831
curve_end[1] =30.000000000000000
curve_end[2] =-9.999999999999591
attach face:
face_origin:
point[0]=5.000000000000036
point[1]=30.000000000000000
point[2]=-12.499999999999610
face_dir:
dir_x[0]=1.000000000000000
dir_x[1]=0.000000000000000
dir_x[2]=0.000000000000000
dir_y[0]=0.000000000000000
dir_y[1]=0.000000000000000
dir_y[2]=-1.000000000000000
dir_z[0]=0.000000000000000
dir_z[1]=1.000000000000000
dir_z[2]=0.000000000000000
#
operation_name:EXTRUDED
operation_parameter:
bool_statu=7
so_creatio=1
tap_angle=0.000000
start_dist=0.000000
end_distan=18.000000
first_offs=0.000000
secon_offs=0.000000
direction:
d_x: 0.000000
d_y: -1.000000
d_z: 0.000000
extrude_curves:
num_curves: 4
curver_type[0]=5
curve_center[0]=-5.000000000000089
curve_center[1]=30.000000000000000
curve_center[2]=-22.499999999999540
star_angle =1.5707963267948966
end_angle =4.7123889803846906
radius =2.500000000000000
start_point_x =-5.000000000000089
start_point_z =30.000000000000000
start_point_x =-24.999999999999540
end_point_x =-5.000000000000071
end_point_z =30.000000000000000
end_point_x =-19.999999999999540
curver_type[1]=3
curve_start[0]=-5.000000000000098
curve_start[1]=30.000000000000000
curve_start[2]=-24.999999999999540
curve_end[0] =4.999999999999902
curve_end[1] =30.000000000000000
curve_end[2] =-24.999999999999540
curver_type[2]=5
curve_center[0]=4.999999999999911
curve_center[1]=30.000000000000000
curve_center[2]=-22.499999999999540
star_angle =-1.5707963267948970
end_angle =1.5707963267948970
radius =2.500000000000000
start_point_x =4.999999999999902
start_point_z =30.000000000000000
start_point_x =-19.999999999999540
end_point_x =4.999999999999902
end_point_z =30.000000000000000
end_point_x =-24.999999999999540
curver_type[3]=3
curve_start[0]=-5.000000000000089
curve_start[1]=30.000000000000000
curve_start[2]=-19.999999999999500
curve_end[0] =4.999999999999911
curve_end[1] =30.000000000000000
curve_end[2] =-19.999999999999500
attach face:
face_origin:
point[0]=4.999999999999911
point[1]=30.000000000000000
point[2]=-22.499999999999540
face_dir:
dir_x[0]=1.000000000000000
dir_x[1]=0.000000000000000
dir_x[2]=0.000000000000000
dir_y[0]=0.000000000000000
dir_y[1]=0.000000000000000
dir_y[2]=-1.000000000000000
dir_z[0]=0.000000000000000
dir_z[1]=1.000000000000000
dir_z[2]=0.000000000000000
#
operation_name:CIRCULAR_ARRAY
parent feature:
num of parent feature=4
INSTANCE[0]/EXTRUDED
Cut
parent_direction:
X: 0.000000
Y: -1.000000
Z: 0.000000
parent_location:
X: -5.000000
Y: 30.000000
Z: 5.000000
INSTANCE[1]/EXTRUDED
Cut
parent_direction:
X: 0.000000
Y: -1.000000
Z: 0.000000
parent_location:
X: -5.000000
Y: 30.000000
Z: -5.000000
INSTANCE[2]/EXTRUDED(4)
Cut
parent_direction:
X: 0.000000
Y: -1.000000
Z: 0.000000
parent_location:
X: -5.000000
Y: 30.000000
Z: -15.000000
INSTANCE[3]/EXTRUDED
Cut
parent_direction:
X: 0.000000
Y: -1.000000
Z: 0.000000
parent_location:
X: -5.000000
Y: 30.000000
Z: -25.000000
array csys:
array_origin:
o_x=0.000000
o_y=0.000000
o_z=-35.000000
array_axis:
dir_x[0]=0.000000
dir_x[1]=0.000000
dir_x[2]=1.000000
array param:
number: p15= 4
angle: p16=90.000000
#

```



```

operation_name:EXTRUDED
operation_parameter:
  bool_statu=7
  so_creatio=1
  tap_angle=0.000000
  start_dist=0.000000
  end_distan=3.000000
  first_offs=0.000000
  secon_offs=0.000000
direction:
  d_x: 0.000000
  d_y: 0.000000
  d_z: 1.000000
extrude_curves:
  num_curves: 2
  curver_type[0]=5
  curve_center[0]=0.00000000000000032
  curve_center[1]=0.0000000000000000
  curve_center[2]=-29.9999999999999610
  star_angle =0.0000000000000000
  end_angle  =6.2831853071795853
  radius     =31.5000000000000000
  start_point_x  =-31.499999999999960
  start_point_z  =0.0000000000000000
  start_point_x  =-29.9999999999999610
  end_point_x    =-31.499999999999960
  end_point_z    =-0.00000000000000357
  end_point_x    =-29.9999999999999610
  curver_type[1]=5
  curve_center[0]=0.00000000000000032
  curve_center[1]=0.0000000000000000
  curve_center[2]=-29.9999999999999610
  star_angle =0.0000000000000000
  end_angle  =6.2831853071795853
  radius     =28.5000000000000000
  start_point_x  =-28.499999999999960
  start_point_z  =0.0000000000000000
  start_point_x  =-29.9999999999999610
  end_point_x    =-28.499999999999960
  end_point_z    =-0.00000000000000323
  end_point_x    =-29.9999999999999610
attach face:
face_origin:
  point[0]=0.00000000000000032
  point[1]=0.0000000000000000
  point[2]=-29.9999999999999610
face_dir:
  dir_x[0]=-1.0000000000000000
  dir_x[1]=0.0000000000000000
  dir_x[2]=0.0000000000000000
  dir_y[0]=0.0000000000000000
  dir_y[1]=1.0000000000000000
  dir_y[2]=0.0000000000000000
  dir_z[0]=0.0000000000000000
  dir_z[1]=0.0000000000000000
  dir_z[2]=-1.0000000000000000
#
feature_name:CHAMFER
edge_chamfer param:
  chamfer type: 1
  first offsetp26=2
  second offset=2
  angle =45
  num of edges: 1
chamfer_edge[0]:
  edge_type[0]=3002
  vertex_count[0]=0
  point3[0]=-11.838748
  point3[1]=-0.000000
  point3[2]=70.000000
#

```


APPENDIX C

Testing part 3

➤ Part and operations

Unigraphics operation:

Cylinder
Extrude_add
Circular_array
Boss
Boss
Hole
Circular_array
Hole
Hole
Edge_blend
Edge_chamfer

Standard operation:

Cylinder
Extrude_add
Circular_array
Boss
Boss
Hole
Circular_array
Hole
Hole
Edge_blend
Edge_chamfer

SolidWorks operation:

Extrude_base
Extrude_boss
Circular_array
Extrude_boss
Extrude_boss
Extrude_cut
Circular_array
Extrude_cut
Extrude_cut
Extrude_cut
Edge_fillet
Edge_chamfer

➤ Modeling process based on standard operations

Testing part 4

➤ Part and operations

Unigraphics operation:
Extrude_base
Extrude_cut
Extrude_cut
Extrude_add

Standard operation:
Extrude_base
Extrude_cut
Extrude_cut
Extrude_add

SolidWorks operation:
Extrude_base
Extrude_cut
Extrude_cut
Extrude_boss

➤ Modeling process based on standard operations

Testing part 5

➤ Part and operations

Unigraphics operation:

Block
Pad
Pad
Boss
Hole
Hole
Hole
Hole
Hole
Hole
Hole
Circular_array
Edge_blend
Edge_blend
Edge_chamfer

Standard operation:

Block
Pad
Pad
Boss
Hole
Hole
Hole
Hole
Hole
Hole
Hole
Circular_array
Edge_blend
Edge_blend
Edge_chamfer

SolidWorks operation:

Extrude_base
Extrude_boss
Extrude_boss
Extrude_boss
Extrude_cut
Extrude_cut
Extrude_cut
Extrude_cut
Extrude_cut
Extrude_cut
Extrude_cut
Circular_array
Edge_fillet
Edge_fillet
Edge_chamfer

➤ Modeling process based on standard operations

Testing part 6

➤ Part and operations

Unigraphics operation:
Extrude_base
Extrude_add
Slot
Boss
Hole
Extrude_add
Edge_blend
Edge_blend
Hole
Hole
Extrude_cut

Standard operation:
Extrude_base
Extrude_add
Extrude_cut
Boss
Hole
Extrude_add
Edge_blend
Edge_blend
Hole
Hole
Extrude_cut

SolidWorks operation:
Extrude_base
Extrude_boss
Extrude_cut
Extrude_boss
Extrude_cut
Extrude_boss
Edge_fillet
Edge_fillet
Extrude_cut
Extrude_cut
Extrude_cut

➤ Modeling process based on standard operations

Testing part 7

➤ Part and operations

Unigraphics operation:

Extrude_base
Extrude_add
Extrude_cut
Extrude_add
Extrude_add
Edge_blend
Hollow
Extrude_add
Boss
Boss
Edge_blend

Standard operation:

Extrude_base
Extrude_add
Extrude_cut
Extrude_add
Extrude_add
Edge_blend
Hollow
Extrude_add
Boss
Boss
Edge_blend

SolidWorks operation:

Extrude_base
Extrude_boss
Extrude_cut
Extrude_boss
Extrude_boss
Edge_fillet
Shell
Extrude_boss
Extrude_boss
Extrude_boss
Edge_fillet

➤ Modeling process based on standard operations

Testing part 8

➤ Part and operations

Unigraphics operation:
Revolve_base
Extrude_add
Extrude_add
Edge_blend
Extrude_cut
Extrude_cut
Extrude_cut

Standard operation:
Revolve_base
Extrude_add
Extrude_add
Edge_blend
Extrude_cut
Extrude_cut
Extrude_cut

SolidWorks operation:
Revolve_base
Extrude_boss
Extrude_boss
Edge_fillet
Extrude_cut
Extrude_cut
Extrude_cut

➤ Modeling process based on standard operations