COLLABORATIVE SESSION MANAGEMENT IN DISTRIBUTED ENGINEERING DESIGN AND ANALYSIS ENVIRONMENT

SUN DONGWEI

(*B. Eng. Beijing University of Posts and Telecommunications*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgments

I would like to express my most sincere gratitude to my supervisor Dr. Liu Zhejie for his invaluable guidance, patience and support over the entire course of my research project. Without his constant support and advice, my completion of this thesis research would not be possible.

I would like to extend my gratitude to Dr. Zhao Jinmin for providing helpful advice and constructive suggestions for my research. My appreciation also goes to all the staff in Data Storage Institute, for their kind assistance during the graduate research.

In addition, I want to thank my friends and fellow students. I am especially grateful to Mr. Li Jiangtao, who has been kindly sharing his knowledge and research experiences with me. Special thank is extended to Ms. Liao Rong for always inspiring me and helping me in difficult times.

Finally, I would like to thank my parents, Sun Hui and Zhao Guilin, for their unconditional love and support in my life.

Contents

Summary

Global competition among manufacturing enterprises has brought in great change in product realization. Companies are embracing Collaborative Product Commerce (CPC), the emerging collaboration-oriented philosophy, to manage product quality, cost and time-to-market in line with the global trend of competition in manufacturing between supply chains. In the CPC environment, collaborative product design becomes the critical phase has a vital impact on the efficiency of the whole collaborative commerce. Rapid advances in computer networks and information technology provide an infrastructure to support the distributed and collaborative product design. According to the nature of products and collaboration requirements, collaborative sessions are established to provide real-time interactions and information sharing between participating collaborators.

The organization and management of a collaborative session in a distributed collaborative design environment have attracted attention from both commercial software developers and academia. However, the research efforts in general tend to focus more on facilitating the collaborations in Computer-Aided Design (CAD) fields without involving the integration of Computer-Aided Engineering (CAE) capabilities, which is a crucial step at the design stage for evaluating the product performance and behaviors.

This thesis presents a distributed collaborative CAD/CAE framework to support not only CAD collaborations but also CAE collaborations. Based on .NET and J2EE technology, the framework has seamlessly wrapped the workflow system and the product design system to manage collaborative sessions.

A workflow-driven mechanism for organizing collaborative sessions has been introduced. During the execution of the product design process, collaborative sessions are managed by a workflow model in which all the task specifications are defined. Ultimately, the product design workflow model is expected to improve the flexibility of product development by effectively organizing collaborative sessions.

A centralized coordination mechanism has been proposed for the management of synchronous collaborative sessions. Under this mechanism, a multi-thread synchronization scheme for collaboration process has been proposed to realize efficient real-time interaction. Such synchronization scheme can provide efficient and effective synchronization of operation, initial representation, and session status. The proposed framework can provide a stable platform to realize efficient synchronized engineering design and analysis.

A collaborative design case of hard disk spindle motor is presented to demonstrate the effectiveness of the proposed framework, which has integrated CAD and CAE functionalities in a distributed collaborative design environment, to support the product development process and the collaborative session management. The development of the framework has special reference to data storage industry which is globalized and has significant presence in Singapore.

List of Tables

List of Figures

Chapter 1

Introduction

Confronted with global competition and rapidly changing customer requirements, manufacturers face an increasingly arduous task in developing new products. Product development is becoming more reliant on geographically dispersed, multi-disciplinary designers during design, manufacturing, and delivery processes. To improve collaboration in product development, companies are embracing Collaborative Product Commerce (CPC), which is an emerging design philosophy that enables companies to be more responsive to the market needs. Advances in computer networks and information technology have enabled global and distributed design teams to more effectively communicate, collaborate, obtain and exchange a wide range of information and design resources throughout product development cycle using CPC solutions. Collaborative product development is going to receive a lot more attention, because the activities of the design process determine both product competitiveness and cost in collaborative commerce. By making the entire collaborative product design process work more effectively, manufacturers are taking a vantage position to manage product quality, cost and time-to-market.

The research effort presented in this thesis is to develop a distributed collaborative engineering design and analysis framework. A workflow-driven mechanism has been introduced to organize collaborative sessions that facilitate collaborative product design activities in a distributed environment. In this research work, the synchronous collaborative session has been studied and a synchronization scheme has been proposed to improve collaboration efficiency.

1.1 Background

1.1.1 Overview

Rapid advances in information technology are creating new opportunities for manufacturing companies to revitalize their competitive strategies. Many companies are now embracing CPC, which leverages on the recent developments in information technologies [1, 2, 3].

The Aberdeen Group [1] defines CPC as " ... a class of software and services that uses Internet technologies to permit individual to collaboratively develop, build, and manage product throughout the entire lifecycle". As Aberdeen defines it, CPC delivers two primary business benefits. First, it improves product quality and capability by connecting "islands of product knowledge" into a single, extended experience base; and, it collapses time and distance by using the Internet to speed time-to-market. Second, CPC tries to foster collaboration between contributors from internal and external organizations throughout the product lifecycle (Figure 1.1).

From Figure 1.1, we can find that product design is the basis of collaborative commerce. Nearly all of the competitive characteristics of a product are determined at its design time. Hence, the product design phase has a high impact on the efficiency of the whole CPC solutions. Its design in technical and organizational aspects affects time, cost and quality of the product development. The measure of coordination and synchronization in a collaboration process significantly influences the collaboration efficiency. Therefore, collaborative product design should be studied.

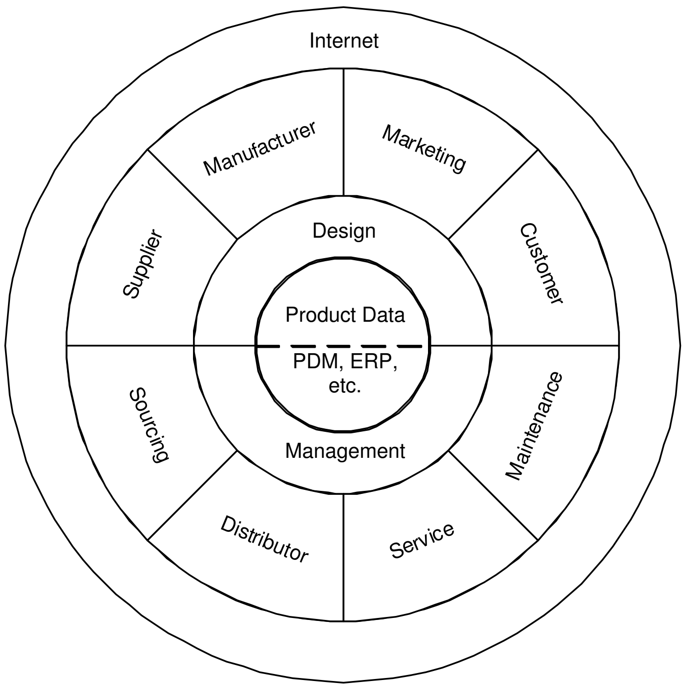


Figure 1.1: CPC solutions provide an aggregate view of product development

1.1.2 Collaborative Engineering Design and Analysis

- **Computer Aided Engineering Design and Analysis**

Computer Aided Design (CAD), emerged in the early 1960s, relates to the use of computers to assist the design process and make the design process more efficient. Specialized CAD programs support for various types of design such as architectural, engineering, electronics, etc. CAD programs usually allow a structure to be built up from some re-usable 2D or 3D templates. It is normally possible to generate engineering drawings to allow the final physical product to be manufactured.

Computer-Aided Engineering (CAE) generally relies on discretization of geometry, description of part attributes/properties and physical conditions, and solution of the large scale simultaneous algebraic equations resulting from specific numerical algorithms. ADAMS [4] and ANSYS [5] are some of the CAE tools. The CAE tools are typically implemented with a stronger emphasis on the detail design phase in the product development process when iterative computer simulations are extensively used to find design optimum and when the major design parameters are verified.

The rapid development and implementation of collaborative tools allow CAD to be used in a greater part of the development process. However, integrated analysis is only possible when CAD presentation of the digital product can be utilized to model the relevant physical processes, for instance, to create finite element models for structural analysis, dynamic models for simulation of motion or for performing Computational Fluid Dynamics (CFD) simulation.

- **Collaborative Engineering Design and Analysis**

With the globalization of manufacturing and product development activities, enterprises are strategically distributing their design and manufacturing activities in different regions to remain competitive. At the same time, a greater focus on outsourcing has spawned new business partners and closer relationships with external suppliers. As a result, collaborations across enterprise boundaries and geographical or disciplinary barriers are commonly practiced throughout the product lifecycle in some of the industry segments, such as data storage industry. Collaboration is particularly vital for product design since this upstream activity in the product lifecycle has a decisive impact on the success of the product.

Collaborative engineering design and analysis can be regarded as a process in which a group of designers jointly design a product model and evaluate its performance. This would include the disparate functions in design, assembly, evaluation and those in suppliers and customers (Figure 1.2). The benefits of collaborative engineering design and analysis might include optimizing the product functions, minimizing assembly costs, eliminating unnecessary engineering change effort and expense, etc. Since a distributed collaborative design team often works in parallel and independently with different engineering tools distributed in separate locations, even across various time zones around the world, the resulting design process may then be called distributed collaborative design [6].

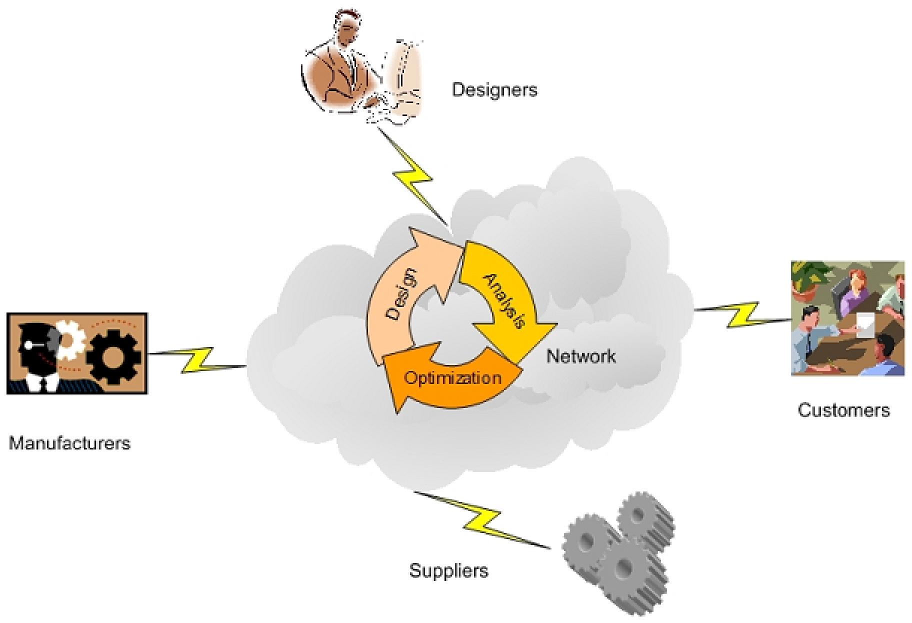


Figure 1.2: Collaborative engineering design and analysis

In the development of complex products, the accomplishment of a design process usually needs information feedbacks from manufacturers, suppliers and customers who are located in distributed areas. Under the circumstances, the traditional sequential design process becomes inflexible and time-consuming, since all these information feedbacks are performed by human interactions. Distributed collaborative design can solve this problem. In a distributed collaborative design environment, people from different fields are brought together to discuss on the product model and evaluate product performance. Therefore, the constraints and conflicts can be detected in the early design stage and the design efficiency can be improved considerably.

In order to carry out distributed collaborative design effectively, network-based sessions are established in a distributed collaborative design environment to support reliable collaboration between geographically dispersed engineering teams. In a collaborative session, different engineers can share the common data and communicate with each other through conferencing tools, such as email, instant messaging tools, etc.

Traditionally, a session is the term refering to a process in which a collection of users connect from various locations to work together on shared data or use conferencing tools to communicate ideas [7]. However, collaborative product design activities include asynchronous activities as well as synchronous activities. Dependencies exist between sequential activities, that is, even if one may work as an individual to perform an asynchronous collaboration activity, he still works on shared resources which may affect other collaboration activities. In addition, the functions of traditional network-based sessions have to be extended in order to facilitate both design and analysis activities, such as providing co-modeling, visualization of meshing result and engineering data. Thus, the definition of session for distributed collaborative product design may be extended as:

The process in which multi-discipline designers, who may be from geographically dispersed locations, work together to design product or analyze engineering results, synchronously or asynchronously, with the help of collaboration tools.

Synchronous design means that designers carry out the same design task in the same workplace. They work on the product model concurrently, such as co-modelling, co-analysis. Asynchronous design means that designers carry out different design tasks in different workplace. They work on the different part of the product model and do not need to be at the same pace.

1.2 Problem Statement

- **Distributed Collaborative CAD/CAE Framework**

The existing product design systems provide an effective tool for product geometric modeling. However, most of them have limited co-modeling functions for distributed collaborative design. In addition, they lack the capabilities of integrating CAE, which is a crucial step at the design stage for evaluating the product performance and behaviors. Although some studies have reported on integration with respect to CAD/CAE functions, they did not provide an adequate integrated environment for overall effective product design. With more and more product complexity, to reduce time-to-market and lower the cost of product development, it is necessary to evaluate product performance in product design stage. Meanwhile, Microsoft .NET Remoting provides much more complex functionalities than traditional component-based technologies, and is becoming a powerful tool for the development of distributed computing solutions.

It is therefor one of the targets of this research to develop a distributed collaborative CAD/CAE framework based on Microsoft .NET technology that can meet the need of integration of CAD and CAE. The system is expected to result in reduced development time, cost saving, improved product quality and faster response to the customer requirements.

- **Collaborative Session Management**

The organization and management of collaborative sessions in a distributed engineering design environment have attracted attention from both commercial software developers and academia. However the research efforts to-date tend to focus more on facilitating the collaborations of CAD without involving the integration of the CAE capabilities.

Collaborative product development has necessitated distributed workflow management to be adopted to manage and monitor distributed interactive activities in a product lifecycle. By facilitating the interoperation of mechanisms in a heterogeneous environment, distributed workflow management can support both design and adaption to the dynamic changes of resources needs and availability in a distributed environment. It can be envisaged that the management system for distributed workflow can be leveraged to play an important role in managing collaborative sessions if it can be integrated with product design system and its functions extended to dynamically define collaborative sessions.

In this research work, a workflow management system has been integrated with the product design system. A workflow-driven collaborative session management mechanism is introduced in an attempt to achieve the following benefits in such a distributed collaborative engineering design and analysis environment:

First, it can automate the product design process. Real-time monitoring of collaborative sessions as well as auditing of product design processes become possible. This can reduce costs, streamline processes and result in better session management and tracking. Second, through deploying workflow model, the reusability of activity implementations in product design process can be achieved. Finally, the distributed activities that may take place in heterogeneous environments are well organized and the capabilities of disparate applications are well integrated.

- **Synchronization in Synchronous Collaborative Session**

In a collaborative session, synchronization is to synchronize view, data representation, and operations. It is one of the most critical problems involved in the distributed engineering design environment. The problem becomes more challenging because of the integration of CAE capabilities. Synchronization between clients is crucial not only for design processes but also for pre/post-processes during the product performance evaluation period. A good synchronization scheme can ensure efficient and effective collaborative engineering design and analysis processes.

In this thesis, the synchronous collaborative session management is studied in detail. A new synchronization scheme for synchronous collaboration is proposed. In this scheme, the proposed framework can provide a stable platform to realize efficient synchronized engineering design and analysis.

1.3 Research Objectives

The objectives of this research work are:

- Develop a distributed collaborative CAD/CAE framework for distributed collaborative engineering design and analysis based on Microsoft .NET technology.

- Investigate product design process and introduce a workflow-driven mechanism to manage collaborative sessions in the framework.
- Study the synchronous collaborative session, propose and implement a synchronization scheme for such session.

1.4 Structure of Thesis

The remainder of this thesis is organized as follows:

Chapter 2 - It presents a literature survey of the subject in the aspects of technologies, commercial tools and academic researches.

Chapter 3 - It introduces the framework for distributed collaborative design and analysis. It presents the software architecture as well as the functionality of this framework.

Chapter 4 - It discusses the collaborative session management in the proposed framework. It investigates the product design process and illustrates the workflow-driven collaborative session management mechanism. It also studies the synchronous collaborative session and proposes a synchronization scheme for real-time interaction in the synchronous collaboration process.

Chapter 5 - It presents a case study relating to design of a hard disk spindle motor. It depicts how the collaborative sessions are driven by a workflow model, as well as the coordination and synchronization flow in synchronous collaboration process.

Chapter 6 - It concludes this research work and suggests future directions in the relevant areas.

Appendix A - It presents the list of publications arising from this thesis.

Appendix B - It presents the list of abbreviations.

Appendix C - It presents the main server side Visual C# code.

Appendix D - It presents the main client side Visual C++ code.

Chapter 2

Review of Developing Technologies and Methodologies

Building up a distributed collaborative environment for product design and managing collaborative sessions in the environment have attracted attentions from both software developers and academia. In this chapter, the major technologies that support distributed collaborative systems are introduced. The key features and collaboration mechanisms of the current commercial tools for collaborative design are summarized. Based on a literature review of the R&D activities in the relevant fields, the developing technology and client-server architecture adopted in this thesis are discussed.

2.1 Technologies to Support Distributed Collaborative System

In a distributed product development process, design tasks are highly interrelated. Technological developments introduced to the distributed collaborative engineering design and analysis system must therefore support the need of interactions. In the following sections, the major technologies that support distributed collaborative systems are introduced.

2.1.1 Traditional Component-based Technologies

Traditional Component-based Technologies include: OMG's Common Object Request Broker Architecture (CORBA) [8], JavaSoft's Java/Remote Method Invocation (Java/RMI) [9], and Microsoft's Distributed Component Object Model (DCOM) [10].

- **CORBA**

CORBA is an open, vendor-independent architecture and infrastructure that computer applications use to work together over networks using the standard Internet Inter-ORB Protocol (IIOP) [8]. CORBA is a standard architecture allows vendors to develop Object Request Broker (ORB) products that support application portability and interoperability across different programming languages, hardware platforms, operating systems, and ORB implementations.

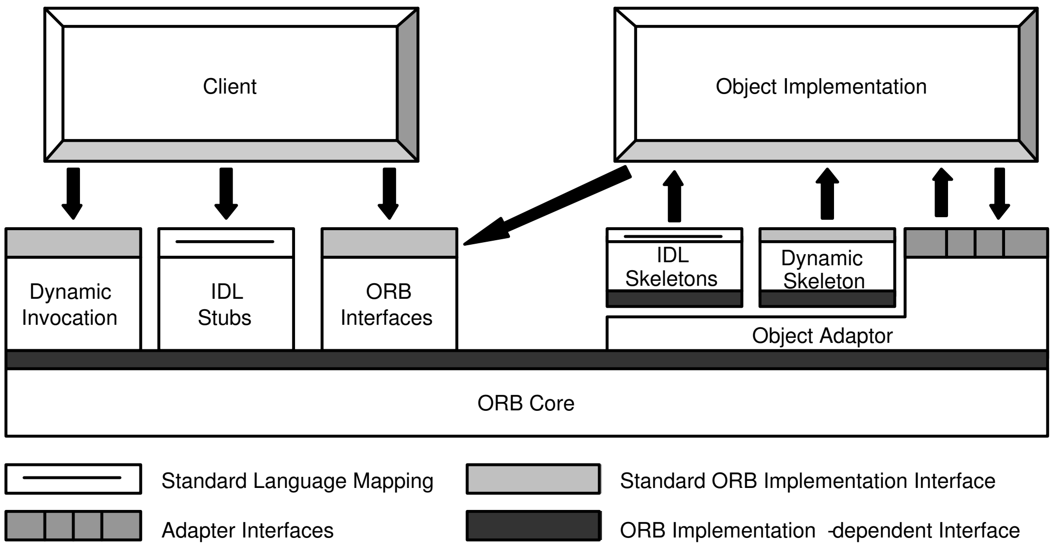


Figure 2.1: CORBA ORB architecture

Figure 2.1 shows the structure of CORBA ORB. All objects are defined in CORBA using Interface Definition Language (IDL). Language mappings are defined from IDL to programming languages, such as C++, Java, etc. "Using a CORBA-compliant ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call, and is responsible for finding an object that can implement the request, passing it the parameters, invoking its method, and returning the results of the invocation" [8]. CORBA specifies that clients and object implementations can be written in different programming languages and executed on different computer hardware architectures and different operating systems, and that clients do not have to be aware of the details about each other.

- **Java/RMI**

Java Remote Method Invocation (Java/RMI) enables the programmer to create distributed Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts [9]. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

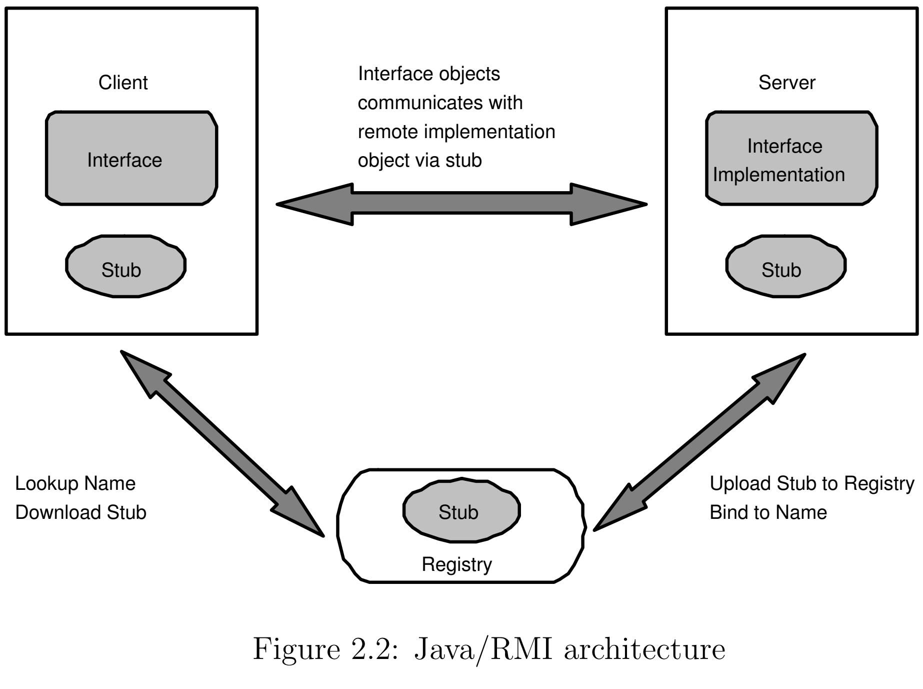


Figure 2.2: Java/RMI architecture

As shown in Figure 2.2, the basic structure of a RMI-based method call involves a client, a server and a registry. To make a call to a remote object, the client first looks up the object it wishes to invoke in the registry. The registry returns a reference to the object on the server, which the client can use to invoke any method that the remote object implements. The client communicates with the remote object via a user-defined interface that is actually implemented by the remote object. The client actually does not deal directly with the remote object at all, but with a code stub that deals with the process of communication between client and server using sockets by default.

- **COM/DCOM/COM+**

Component Object Model (COM) defines an Application Programming Interface (API) to allow components to be created for integration of custom applications, and to allow diverse components to interact. DCOM is an extension to COM to allow network-based component interaction. While COM processes can run on the same machine but in different address spaces, the DCOM extension allows processes to be spread across a network. With DCOM, components operating on a variety of platforms can interact, as long as DCOM is available within the environment. COM+ integrates Microsoft Transaction Server (MTS) services and message queuing into COM, and makes COM programming easier through a closer integration with Microsoft languages.

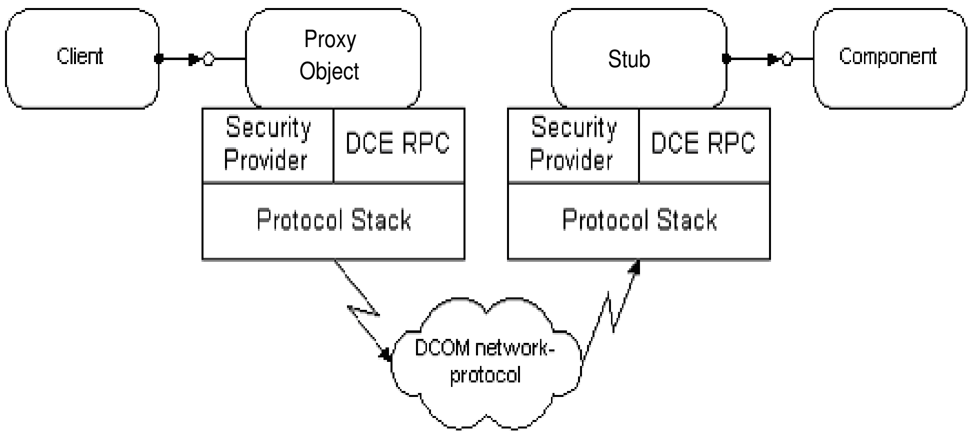


Figure 2.3: DCOM architecture

The DCOM client calls the interfaces of the server through the proxy, which marshals the parameters and passes them to the server stub. The stub unmarshals the parameters and makes the actual call inside the server object. When the call completes, the stub marshals return values and passes them to the proxy, which in turn returns them to the client. The same proxy/stub mechanism is used when the client and server are on different machines. However, the internal implementation of marshalling and unmarshalling differs depending on whether the client and server operate on the same machine (COM) or on different machines (DCOM). Given an IDL file, the Microsoft IDL compiler can create default proxy and stub code that performs all necessary marshalling and unmarshalling. When client and component reside on different machines, DCOM simply replaces the local interprocess communication with a network protocol.

Figure 2.3 shows the overall DCOM architecture: The COM run-time provides object-oriented services to clients and components, and uses Remote Procedure Call (RPC) and the security provider to generate standard network packets that conform to the DCOM wire-protocol standard.

2.1.2 State-of-the-art Technology - .NET Remoting

The Microsoft .NET [11] strategy was presented by Microsoft officials to the rest of the world in June 2000. Microsoft .NET brings a new model for distributed applications, as a successor to DCOM. The .NET Remoting offers far greater flexibility and extensibility over DCOM.

Microsoft .NET Remoting provides a framework that allows objects to interact with one another across application domains. The framework provides a number of services, including activation and lifetime support, as well as communication channels responsible for transporting messages to and from remote applications. Formatters are used for encoding and decoding the messages before they are transported by the channel. Applications can use binary encoding where performance is critical, or eXtensible Markup Language (XML) encoding where interoperability with other remoting frameworks is essential. All XML encoding uses the Simple Object Access Protocol (SOAP) in transporting messages from one application domain to the other. Remoting was designed with security in mind, and a number of hooks are provided that allow security sinks to gain access to the messages, as well as the serialized stream, before the stream is transported over the channel.

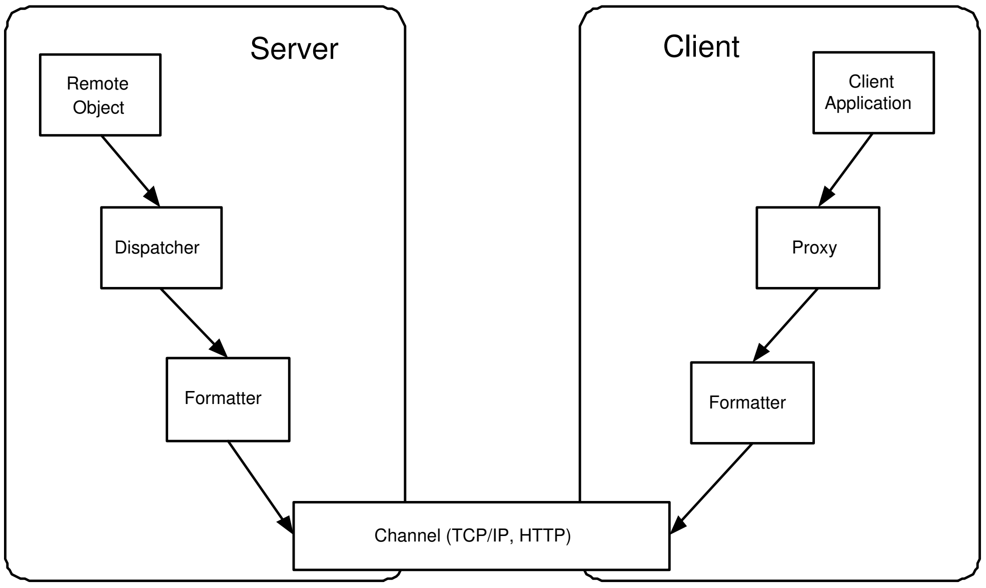


Figure 2.4: .NET Remoting architecture

Figure 2.4 shows an architecture overview of .NET Remoting. The remote object exposes some methods for remote calls. A proxy is created on the client mirroring the remote object in that it exposes the same public methods. The client invokes these methods on the proxy class, and the proxy uses a formatter to format the messages so that they can be sent across the network. The network transport is defined by the channel. On the server, another formatter unformats received messages, and passes them to dispatcher which calls the methods on the remote object.

The .NET Remoting permits interceptors, or sinks, to be placed at certain points in the flow on the client or server-side to add additional functionalities, such as logging, duplicating calls for reliability reasons, or dynamically finding servers.

2.2 Commercial Tools for Collaborative Engineering Design

In a distributed collaborative design environment, designers bring their own personal viewpoints to the product model [12], interact and reach agreement while sharing common information. Current commercial design software is a relatively new integration of networking and CAD. The applications can support collaboration of the designers working on a common design task, each with specific core competencies, interacting in the design process. These application can be generally classified into two categories [13]:

- **Category I:** Inspection of design models to assist collaborative design activities, such as ConceptWorks [14], eDrawings [16],Centric Innovation Center [17], Hoops Streaming Toolkit [18], Autovue [19], Streamline [20], etc.
- **Category II:** Real-time collaborative design tools to support synchronous co-modeling, such as OneSpace [15], CollabCAD [21], Alibre Design [22], etc.

This section will investigate these two categories of commercially available solutions and their collaborative mechanisms.

2.2.1 Tools Assisting Collaborative Design

The systems in the first category is primarily used to support visualisation, annotation and inspection of design models in a CAD environment. In order to realize high-speed collaborative interaction across networks with the limited bandwidth capability and to enhance the visualization performance, 3D streaming technologies have been adopted to reorganize the large number of meshes from a complex model at different Levels of Details (LODs). These tools can enable designers with faster transmission capabilities to obtain high-resolution images quickly while for slower links to obtain lower resolution images at first, before resolution gradually improves over time.

Table 2.1: Tools that assist collaborative design

Tools	Characteristics Description	Compatible systems
RealityWave ConceptWorks	*Features:* An add-on component of SolidWorks *Functions:* View, markup and message	SolidWorks
SolidWorks eDrawing	Features: A viewer for native or simplified CAD files *Functions:* view, mark-up, measure, hyperlinks, layouts and animation	SolidWorks, AutoCAD, CATIA,Pro/E
Centric Software Innovation Center	*Features:* A platform to provide a digital meeting room for product design *Functions:* View, mark-up, video/audio conferencing, chat	Pro/E, CATIA
Tech Soft Hoops Streaming Toolkit	*Features:* An SDK for reading and/or writing highly compressed HSF files *Functions:* Compression, color support, HSF visulization	Pro/E, IronCAD
Cimmetry Systems Autovue	*Features:* A viewer for part and assembly models, supports viewing printing, plotting and conversion features *Functions:* View, manipulate, measure, mark-up, redline, annotate	CATIA, Pro/E, Autodesk Inventor, AutoCAD, SolidWorks, Solid Edge
Autodesk Streamline	*Feature:* A platform based on the VizStream, collaborates through email and report *Functions:* View, measure, email	Autodesk Inventor, SolidEdge, SolidWorks

Commercial tools under Category I include ConceptWorks [14], eDrawings [16],Centric Innovation Center [17], Hoops Streaming Toolkit [18], Autovue [19], Streamline [20], etc. The key characteristics of some tools under this category are summarized in Table 2.1.

The tools in Category I can support real-time produce design review process which is important when performing a stage discussion or doing a customer survey for new products. However, since most of these tools are based on simplified CAD files, real-time collaborative design, such as co-creation and co-modification, cannot be effectively supported. Therefore, they can only serve as supporting tools.

2.2.2 Tools Supporting Real-time Collaborative Design

Table 2.2: Tools that support real-time collaborative design

Systems	Collaborative Mechanisms	Function Description
CoCreate OneSpace	*Features:* Dynamically includes data from different CAD systems at the same time. Organizes 2D or 3D project files in a database and helps track of document version and history. *Co-design Session:* Shares individual models through a common workspace. Each session is under the guidance of a session administrator. *Data Sharing:* Entire model is download from server and can be stored and shared through database.	Modeling, View, Mark-up, Net-meeting
CollabCAD	*Features:* Uses OpenCASCADE 3D modelling engine and Jython Client-side Scripting to achieve inter application operability and quicker application development. *Co-Design Session:* Multiple Designers can access 2D or 3D models and work on it concurrently across network. *Data Sharing:* Store and share users' models through a database. IGES, STEP, etc. are used for data exchange.	Modeling, view, chat, video conferencing
Alibre Design	*Features:* The 3D CAD application can support feature-based parametric solid modeling and 2D associative drafting *Co-design Session:* Within a design session, designers can create and modify precise 3D models and 2D drawings simultaneously *Data Sharing:* Through repository.	Modeling, view, chat, mark-up, NetMeeting

The systems in the second category can support real collaborative design. The currently available systems include OneSpace [15], CollabCAD [21], Alibre

Design [22]. The collaborative mechanisms of these systems are summarized in Table 2.2.

The tools in Category II can be used to establish a distributed workspace with effective sharing of detialed design models. However, their collaborative design functionalities are limited and the communication efficiencies of these systems are still quite far from satisfactory. This results in the deficient synchronization among designers when collaborating synchronously. In addition, since engineering analysis is a crucial step in product design process, it is necessary to integrate CAE functionalities in distributed CAD environment. However, these commercial tools are not designed to accommodate such functionalities.

2.3 Research and Development in Related Fields

2.3.1 Historical Overview

The researches on distributed collaborative engineering design can be traced back to the time when Computer Supported Cooperative Work (CSCW) [23] was introduced. Since the early 1990s, researchers have tried to integrate CAD and CAE resources over the network. Most of these earlier researches intended to study the interfaces to share environment, such as Ludwig's (1990) [24] research in which a methodology of integrating CAD and CAE using teleconferencing and messaging tools was described, and Shu's Teledesign system (1992) [25] which intended to examine specific issues relating to viewpoints and pointers in a multi-user 3D environment. The Co-CAD system that was developed by Gisi and Sacchi (1992) [26] was claimed to support real-time collaborative solid modeling for mechanical engineers. However, their approach was largely from a mechanical engineer's perspective and limited to collaboration between two people.

These earlier reported approaches for collaborative design over distance included the use of communication tools such as e-mail, multimedia, shared screen or teleconferencing. However, these GroupWare tools have limited functionalities to support real-time collaborative design. Since the earlier research is constrained by network and computer technologies to a large extend, the earlier systems are quite far from the systems of practical use.

Due to the rapid development in network and computer technologies in the late 1990s, there are new opportunities to improve the collaborative design environment. The impact of network technology on design environments has been perceived, and computer support for collaborative design has grown into a major area of research [27, 28, 29, 30].

CORBA 2.0, published in 1994, can provide an efficient protocol for communication and support standardized language mapping. The NetFEATURE presented by Lee et al. (1999) [31] is based on CORBA ORB for communication. The server, implemented using C++, can provide basic modeling functions, such as solid creation, deletion, etc. The client, implemented using Java applet, can handle the local copy of design models. Java/RMI is designed by people who have years of CORBA experience. Its stable, flexible characteristics attract researchers' attention. Based on Java/RMI technology, Chan et al. (1999) [32] developed a Web-based collaborative modeling system, named CSM. The server has a global model and each client has a local copy of this model. When a designer modifies the model, the modifications are propagated to all other users through server. Locking protocol is adopted to avoid operation conflicts. DCOM, introduced in 1996, makes it possible to create network-based applications built from components. Liu (2000) [33] developed a generic component framework for distributed feature-based design and process planning based on COM/DCOM. Data sharing in such system can be realized using standard format, STEP. Xie. et al. (2003) [34] developed CedSpace using DCOM technology. In CedSpace, engineers can collaboratively discuss on a product model though manipulation, mark-up, and chatting tools. A token ring protocol is deployed in the collaborative design framework to avoid operation conflicts. Only the client who owns the token has the right to manipulate the product model. Sever has a queuing list to handle multiple token requests.

In recent few years, the middleware technologies, such as Java/RMI, CORBA and COM/DCOM, have grown mature and are widely used to develop, integrate and distribute software components in an environment of heterogeneous computers, operating systems, network protocols and programming languages. Researchers began to describe a distributed collaborative engineering design environment from different viewpoints, such as functional object model [35], Web-based model [36] and agent-based model [37]. At the same time, the research focus becomes more specified, like collaborative conceptual design [38, 39], collaborative component design [32, 40], and collaborative assembly design [41, 42].

The emergence of .NET technology has brought a new evolution for distributed collaborative design by providing an internet-based platform of next generation windows services. It is expected that collaborative engineering design system can fully leverage .NET technology to realize effective and efficient collaboration activities. With such motivation, this work is to develop a distributed CAD/CAE framework on the basis of .NET.

2.3.2 Recent Work

In recent years, significant research has focused on the technologies or the infrastructure that can assist product designers in the distributed design environment. Li et al. [43] pointed out that the existing collaborative design tools can be broadly classified into two categories:

- **Manipulation client + modeling workspace (thin-client style, A in Figure 2.5):** Clients are equipped with light-weight data structures. Server maintains a common modeling workplace for all clients.
- **Modeling client + communication server (thick-client style, B in Figure 2.5):** Whole CAD system is implemented at client side. The server

acts as a coordination and information exchanger for all clients.

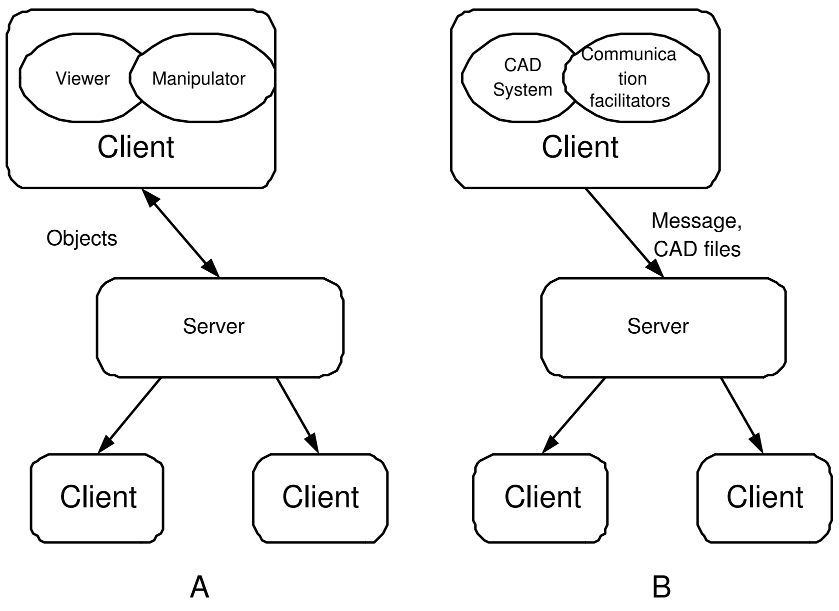


Figure 2.5: Two types of collaborative design tools

The ability of the Web for designers to publish information relevant to the design process has motivated the adoption of the Web as a design collaboration tool. Many researchers have developed Web-based collaborative design systems that belong to the first category. HTML, Java Applets, Active X, VRML, agent, COM/DCOM are widely used for developing the light-weight visualization clients. Table 2.3 summarizes the thin-client style collaborative design systems. The collaboration mechanisms within some typical systems are described in detail.

- **Co-DARFAD**

Co-DARFAD system is a collaborative design system, as introduced by Huang et al. (2001) [51], which is characterized with formal collaborative design process. By standardizing the various design activities, the authors integrated concurrent design and axiomatic design concepts in a unified and structure-oriented automatic design process for mechanical product.

Table 2.3: A summary of thin-client style systems

R&D work	Key features	Development technologies
Pahng et al. (1998) [44]	Multi-server architecture using distributed object technology	Web, CORBA, Java, HTML
Hague et al. (1998) [38]	Localised design agents for conceptual design based on product life cycle information	Agents, Web
Huang et al. (1999) [39]	Web-based collaborative environment using morphological char	Web, HTML, ActiveX
Roy et al. (1999) [45]	Share geometric models in VRML, multi-server architecture	Web, HTML, VRML
Chen and Liang (2000) [46]	A system integrating and sharing engineering information to support CE activities	Web, CORBA, VRML
Shen and Norrie (2000) [37]	A agent architecture ensuring the coordination among design parts and resource agents to support distributed design and manufacturing activities	Agent
Lee et al. (2001) [47]	A Web enabled approach for feature-based modeling in a distributed computing environment	Web, Java
Nidamarthi et al. (2001) [48]	Designers upload and download their CAD files in a server for sharing and exchanging	Web, VRML
Shyamsundar and Gadh (2001) [41]	a new compact assembly representation for Internet-based collaborative assembly design involving clients and subcontractors	Web, Java
Kong et al. (2002) [49]	An Internet-based collaborative system for a press-die design process for automobile manufacturers	CORBA, Java
Ming et al(2002) [50]	A INPROSE system integrating product design, process planning and CNC in a collaborative environment	COM DCOM

Virtual Reality Modeling Language (VRML) is chosen as the common media to allow a product to be viewed interactively across Co-DARFAD system. The visual representation of product model is captured by the client CAD tool and the corresponding VRML file is generated. Other clients can view the VRML based visual product concept, discuss problems and exchange design ideas through web browser.

The advantage of Co-DARFAD system is its flexibility in facilitating collaborative discussions for a whole design process even if clients use different CAD software. However, no special measures are taken to keep reliable synchronization during collaborative design process. In particular, data consistency and concurrent operations in a collaborative session have to be solved by users themselves in order to realize effective product design.

- **DIJA software**

The DIJA software (2003) [52, 53] can be seen as a general framework for web-based CAD systems. Denis et al. proposed a replicated architecture based on a multi-level language. The design information to exchange between two computers is transformed to instructions belongs to the multi-level language. Thus, for the same model, different clients can have different visualizations since they may execute instruction that belongs to different level language. Each client can download its desired data and store locally. Server saves the whole design.

The abstraction levels of DIJA software is implemented using the SDK 1.3 of the Java language and the Java 3D library for visualization. The instructions are stored in a XML format.

DIJA software focuses on the multi-level language based on which server and client can work on the same model in distributed environment. However, synchronization issues in collaboration process are not taken into account. It is only applicable to two applications currently and needs further validation to facilitate collaborative works.

- **e-Assembly**

The e-Assembly system developed by Chen et al. (2004) [54] can support Internet-based collaborative assembly modeling. E-Assembly client is based on

Java Applet and geometric engine is deployed at the server side.

A session server is implemented to provide services with functionalities that enable designers to participate in and exit from a particular collaborative session. A model change event being executed by one e-Assembly client can be published to all other clients through the session server. Also the session server provides message delivery service for all clients during the collaboration process.

The e-Assembly system provides distributed designers with the capability of assembling parts collaboratively in real time and collaboration activities are organized by its session server. However, the functions of session server are limited, for example, online model modification is not supported.

Researches on the thick-client style collaborative design systems are limited. There are several prototype systems that were reported in the literatures. Three typical systems are described as below.

- **CollIDE**

The Collaborative Industrial Design Environment (CollIDE) proposed by Nam and Wright (1998) [55] provides a shared 3D workspace for multiple designers. The main functionality of CollIDE is to provide the shared visualization and accessibility to common 3D objects in a collaborative session. Designers can copy the 3D model that is developed in other 3D modeling system to the shared workspace window. In addition, they can bring geometry from the shared stage to their private stage by selecting it and executing a CollIDE teleportation command. Multiple designers can control a shared camera and see the movement of the camera controlled by others.

CollIDE system has severe restrictions to crucial collaborative design issues. The co-modeling functions are limited. Each designer has to perform geometric modeling in his local CAD system, and then copy to the shared workspace. In addition, synchronization problems during a collaboration process are not addressed.

This will result in delay of operation and congestion of data transmission if conflicts occur.

- **Syco3D**

Synchronous collaborative 3D CAD system (Syco3D) (Nam, and Wright, 2001) [56] uses a shared 3D workspace, called Shared Stage, to facilitate collaborative design activities. Each designer has access to his individual 3D CAD workspace which is linked to the Shared Stage. All designers in a collaborative session see the same 3D view of a 3D virtual workspace and any change in the view is updated instantly. When a designer is manipulating a 3D model in Shared Stage, a locking protocol is adopt to keep other clients from manipulating the same model.

Syco3D system provides a shared workplace for sharing of product information and assembling product. However, the high dependencies between parts of product model are not considered. In addition, the system cannot support co-modification, that is one can only view the part models that others are working on, but he cannot modify these models.

- **WPDSS**

Web Product Design Support System (WPDSS) (Qiang et al. 2001) [57] can support CAD-based collaborative design through the Internet. A client-server architecture is deployed. The server supplies CAD geometry and engineering information to all the clients. Real-time co-design is based on the traditional commercial CAD software among many clients. A Java-based interface is developed to extend the single-location CAD software to a multi-location application through the Internet.

In order to reduce the geometric data transmitted across network, CAD macro is used to record the modification procedures. The micro files generated at one client are broadcasted to other clients through WPDSS server. Clients that receive the macro files will update the geometric model using its local CAD system. A co-modifying monitor is deployed at the server side to monitor the co-modifying session. After one designer's operation, the corresponding macro file is generated and sent to the server in order to update models at other clients. When an update request arrives, the monitor will check if there are other requests to ensure there is one request at one time. Then the macro file is propagated to other clients to update their local copy of product model. The next round operation will not start until the co-modifying monitor receives feedback from all other clients that indicate all clients have updated their models.

The advantages of WPDSS system is that it uses macro files to update operation among different clients. This will improve the efficiency of collaborative design to some extend. However, the synchronization mechanism is not effective. Since a slight operation will generate corresponding macro file, there are many redundant update requests that are not meaningful. In addition, co-modifying monitor has to wait the completion of all clients' updating before starting next operation. This will lead to a large latency of synchronization and the design process might be blocked if the monitor cannot receive feedback due to network congestion.

2.4 Summary

- **Developing Technologies**

Traditional component-based technology, such as the DCOM, CORBA, or Java/RMI, provided reliable, scalable architecture to meet the growing needs of applications.

Though these component-based technologies work very well in an Intranet environment, attempting to use them over the Internet presents two significant problems. First, the technologies do not interoperate. While they all dealt with objects, they disagreed over the details, e.g. lifecycle management, support for constructors, and degree of support for inheritance. Second, and more important, their focus on RPC-style communication typically led to tightly coupled systems built around the explicit invocations of object methods.

Table 2.4: Comparison of developing technologies

Technologies	Interoperating ability	Degree of inheritance	Object invocation
CORBA	Programming languages can be used to code objects only when there are ORB libraries.	Supports multiple inheritance at the interface level	When a client object needs to activate a server object, it binds to a naming or a trader service.
Java/RMI	The objects can only be coded in the Java language.	Supports multiple inheritance at the interface level	When a client object needs a server object reference, it has to do a lookup() on the remote server object's URL name.
DCOM	Since the specification is at the binary level, programming languages like C++ and Java, can be used.	Supports multiple interfaces for objects and uses the QueryInterface() method to navigate among interfaces.	When a client object needs to activate a server object, it can do a CoCreateInstance()
.NET Remoting	Diverse programming languages like C#, C++, Java, and Visual Basic can be used.	Support multiple interfaces. Support COM interfaces. This means that a client proxy dynamically loads multiple server stubs in the remoting layer depending on the number of interfaces being used.	Support for passing objects by value or by reference, callbacks, multiple-object activation and lifecycle management policies.

The .NET Remoting provides an infrastructure for distributed objects. It exposes the full-object semantics of .NET to remote processes using plumbing that is both very flexible and extensible. Compared to traditional component-based technologies, .NET Remoting offers much more complex functionalities, including support for passing objects by value or by reference, callbacks, and multiple-object activation and lifecycle management policies. Table 2.4 gives the comparison of .NET Remoting with other technologies. It shows that .NET remoting can provide more powerful support for product design in a distributed collaborative environment.

In this study, .NET Remoting technology is adopted to implement communication framework in the proposed framework. Based on .NET technology, the framework has proved its ability to realize efficient and effective collaboration in collaborative product design process.

- **Client-Server Architecture**

With the emergence of the Web, the thin-client style has gained much popularity for ease of deployment. The rationale possibly lies in that data consistency and operation synchronization can be easily achieved at a central server. However, several drawbacks still cannot be overcome by thin-client style systems, such as the demand for significant bandwidth for all client applications and the requirement for the user to be online whenever the applications have to be used. While the thick-client style tends to be more robust and capable of handling even the worst network conditions.

Time has changed with the emergence of the new needs in modern industry. Collaborative engineering analysis has been brought forward in order to shorten product development cycle. The ideal system should be capable of supporting collaborative engineering analysis as well as engineering design. To fully participate in a collaborative design process, designers need to be able to, not only exchange general geometric information but also to locate or provide generic analysis services. However, according to the literature review, both the thin-client style and thick-client style support only limited co-design functions through provision of shared information space.

Thus, a new framework has been proposed in this thesis, which is capable of collaborative engineering design and collaborative engineering analysis: modeling

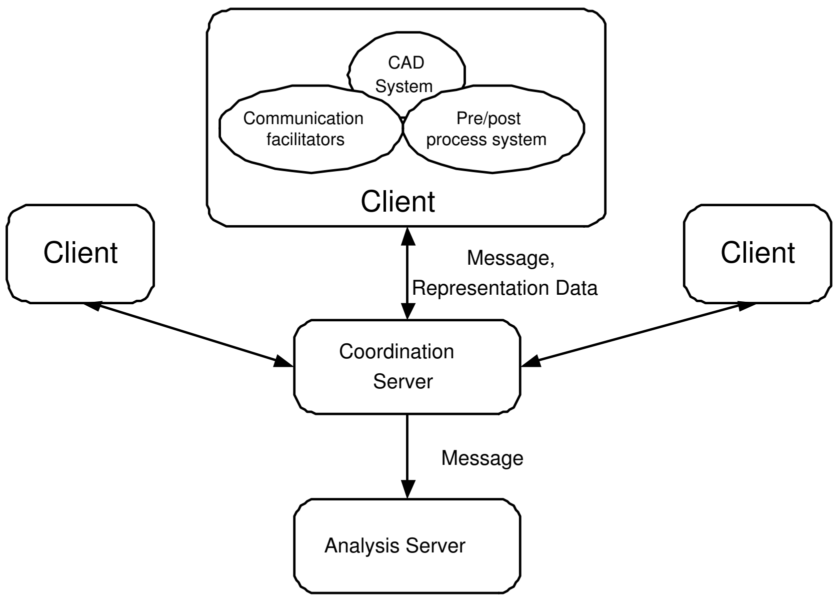


Figure 2.6: New paradigm of collaborative design tool

client + coordination server + analysis server (Figure 2.6). Client is equipped with all necessary CAD facilities. Analysis server provides CAE functionalities for engineering analysis. Coordination server provides communication and coordination for all design and analysis activities.

Since each client is equipped with a stand alone CAD system, most design tasks can be carried out asynchronously. In addition, an enhanced single replication mechanism has been adopted in order to keep data consistency, that is, only one client owns the original data, other clients only have the necessary data for visualization. Synchronization among these clients is achieved by the coordination server. The new framework is expected to have a combination of the advantages of the thin-client style and thick-client style.

Chapter 3

A Distributed Collaborative CAD/CAE Framework

This chapter presents a distributed collaborative CAD/CAE framework, CoCADE, developed during the course of this research work. The framework has seamlessly wrapped a workflow management system and a product design system for the management of collaborative product design sessions which are likely to be dynamic and interdependent.

3.1 Software Architecture

3.1.1 Overview

In order to automate definition of collaborative sessions for design-analysis activities using a workflow model, the architecture for the whole framework shown in Figure 3.1 has been adopted. The figure shows the integrated three-tier architecture of a workflow management system and a product design system.

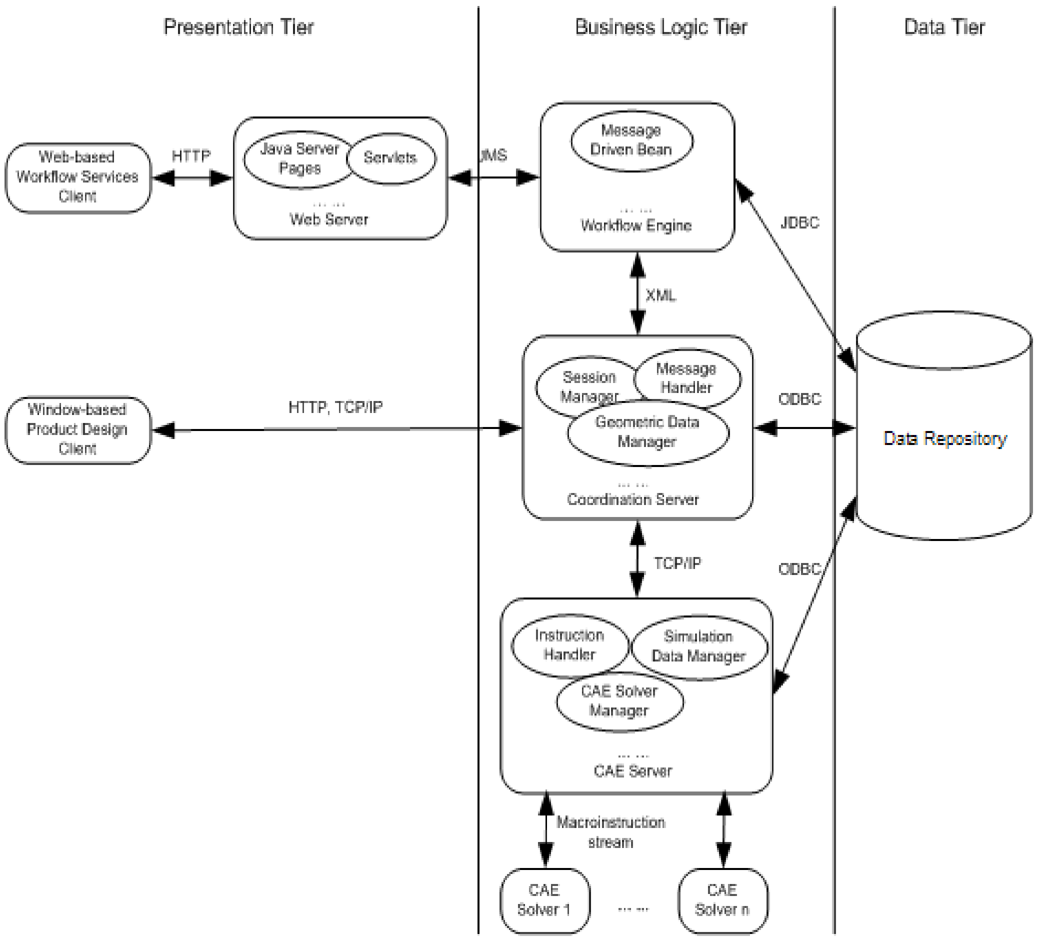


Figure 3.1: Architecture of CoCADE framework

3.1.2 Introduction to Product Geometric Modeling Kernel

The product design system in CoCADE framework adopts ACIS [58], an object-oriented three-dimensional (3D) geometric modeling engine from Spatial Technology Inc., for product geometric modeling. ACIS provides a geometry foundation for 3D modeling application and has the flexibility to adapt or extend for particular application requirements.

Wireframe, surface, and solid modeling are incorporated in ACIS kernel by allowing these alternative representations to coexist naturally in a unified data structure, which is implemented in a hierarchy of C++ classes. Linear and quadric geometry are represented analytically, and Non-Uniform Rational B-splines (NURBS) represent free-form geometry.

ACIS supports two types of file format: SAB and SAT. Both of these two formats contain all necessary model information which can be accessed by applications that are not based on ACIS. The difference between these two file formats is that the data is stored in binary form in SAB files and in ASCII form in SAT files.

Several engineering design systems were implemented on the basis of ACIS geometric kernel, such as AutoCAD [59], IronCAD [60] etc. In this research work, ACIS geometric kernel 9.0 is adopted to enable product geometric modeling by product design client.

3.1.3 Introduction to Workflow

The workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal.

Conventionally, business processes are implemented by hard-coding business process related aspects such as control and data flow into the software systems that are hard to modify and maintain. Workflow management is a technology that addresses these problems. The basic idea in workflow management is to capture formal descriptions of business work process and to support the automatic enactment of the processes based on these descriptions.

Workflow Management System(WfMS) is the software system that supports workflow management. As defined by Workflow Management Coalition (WfMC)[61], WfMS is "a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants, and, where required, invoke the use of IT tools and applications".

A WfMS consists of two main functional components: a build-time component and a runtime component. The build-time component provides support for the development and persistent storage of workflow types. It offers the workflow modeler a workflow modeling language in which workflow types can be expressed together with appropriate tools, such as editors, browsers, and parsers/compilers. Besides workflow modeling, the build-time component also supports organizational modeling, which includes the specification of information about processing entities. For instance, it has to be specified with activities provided by the processing entities. Furthermore, organizational relationships among processing entities may have to be defined in order to enable the specification of activity assignment to processing entities based on organizational relationships. Besides the aforementioned functionalities, the build-time component may provide additional facilities to simulate workflow executions and analyze workflow types.

The runtime component supports the creation and enactment of workflows according to the workflow types created with the build-time component. During the workflow enactment, the runtime component interacts with the processing entities in order to ensure that the workflows are executed as prescribed by the corresponding workflow types. WfMS usually provides monitoring tools that allow the workflow administrator to keep track of the execution progress workflows. Also, WfMS typically maintains logs about workflow executions that can be queried and analyzed in various ways in order to validate workflow types, identify bottlenecks, etc.

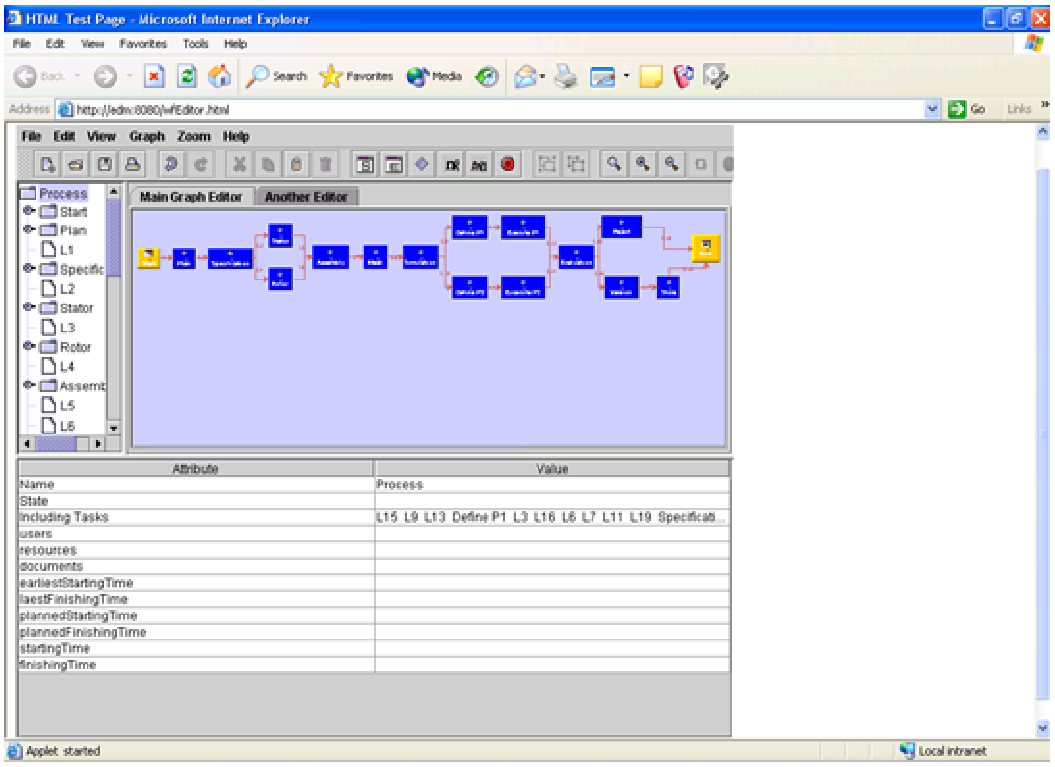
3.1.4 Introduction to Coordination Modes

Generally, there are two modes for coordination activities, which are parallel and sequential.

- **Parallel mode:** In a parallel mode, supposing that a product can be divided into enough number of parts, each individual approaches a part of product. Each part are connected with interface which can be predefined. There may be information provided from each member to the group on the status of his or her progress.

- **Sequential mode:** In a sequential mode, the group imposes phases on the problem solving process that must be undertaken in a sequential manner by all the members of the group. This mode is usually used when assembling each part to one product. Each member can discuss problems and make some changes to his or her own part in the product view which includes all the parts of the product.

Normally, a product design process consists of discrete design tasks. Thus the parallel mode is inevitable to be the main mode for the design process. The sequential mode is also needed when assembling objects or analyzing simulation results.


HTML Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://edm:8080/wfEditor.html
Go Links
File Edit View Graph Zoom Help
Main Graph Editor
Another Editor
Process
Start
Plan
L1
Specific
L2
Stator
L3
Rotor
L4
Assemb
L5
L6
Applet started
Local intranet

Attribute	Value
Name	Process
State	
Including Tasks	L15 L9 L13 Define P1 L3 L16 L6 L7 L11 L19 Specificati...
users	
resources	
documents	
earliestStartingTime	
laestFinishingTime	
plannedStartingTime	
plannedFinishingTime	
startingTime	
finishingTime	

Figure 3.2: Web-based workflow services client

3.2 Product Design Workflow Management System

Considering the context where the workflow resides, the most common scenario is that the application software (including product design client, CAE solver etc.) is running on heterogeneous platforms. To enable all the participants to access a workflow service easily and conveniently, a thin client which is based on web browser is deployed to provide a main interface between the user and the workflow server, as what is shown in the system architecture (Figure 3.1). Figure 3.2 shows the Web-based workflow services client.

In order to model product design workflow, every activity can be regarded as a task in the workflow model. each task consists of state, users, resources, documents, and time requirement, etc. During the execution of the process, all these composites of the task can be changed dynamically. At the same time, the connectors between different tasks represent the relationships between activities. It can be shown in Figure 3.3.

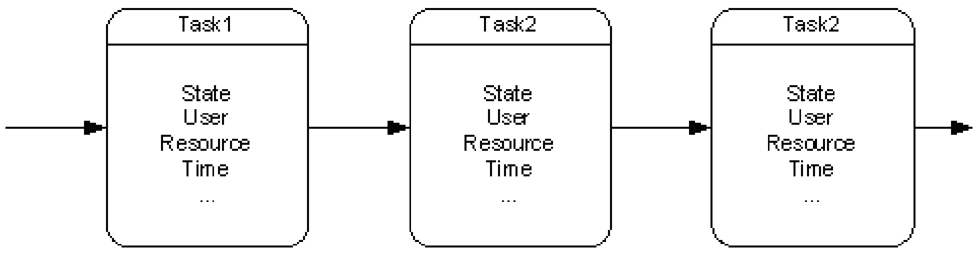
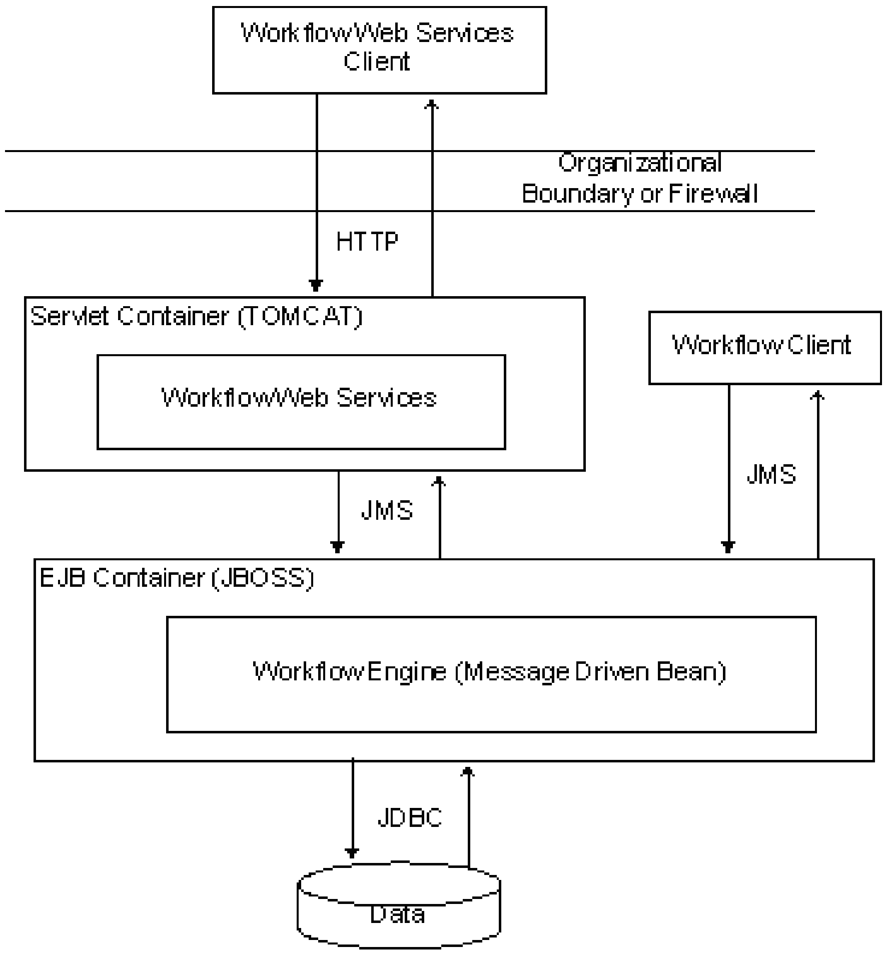


Figure 3.3: Task model of product design workflow

In a product design workflow model, all the tasks are message-driven. A workflow engine that is the core of the workflow management system is responsible for the explanation and execution of the messages. It is built on Enterprise Java Beans (EJB) technology, which is the server-side component architecture for the J2EE, to enable distributed, transactional, secure and portable development of product design workflow. Figure 3.4 shows the deployment view of the workflow



Workflow Web Services Client
Organizational Boundary or Firewall
HTTP
Servlet Container (TOMCAT)
Workflow Web Services
Workflow Client
JMS
JMS
EJB Container (JBOSS)
Workflow Engine (Message Driven Bean)
JDBC
Data

Figure 3.4: Deployment of workflow management system

management system. the workflow engine is a Message Driven Bean (MDB) that runs at EJB container. The advantage of MDB is that they can be pooled and load-balanced to boost for scalability. The workflow engine listens for workflow requests, services them and returns responses. Communication between the client and the MDB is synchronized over Java Message Services (JMS). In this system, the workflow engine is deployed in the Jboss EJB container and workflow web services are deployed in Tomcat.

3.3 Product Design System

Product design system consists of product design client, coordination server, CAE server, and product database. The overall system architecture including general information flow and its relationship with product design workflow system are shown in Figure 3.5.

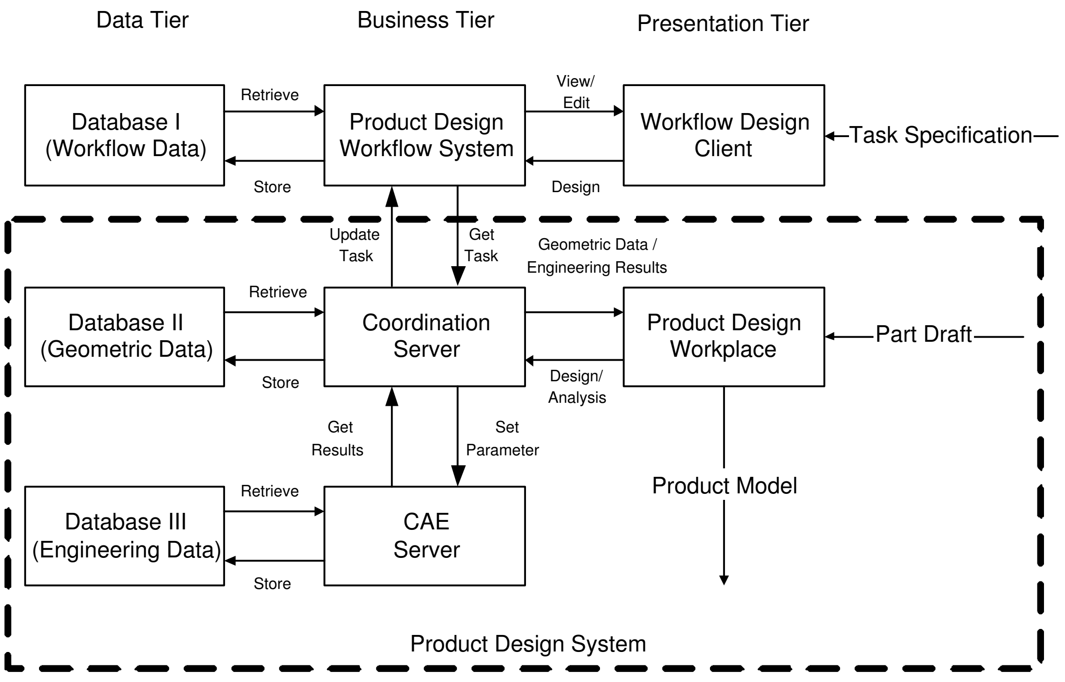


Figure 3.5: General structure of the product design system

Figure 3.6 describes a typical product design flow in product design system. Product design client gets task information from product design workflow system. Then the collaborative session is established to carry out the product design task. Having finished product design process, designers can discuss and decide the data preparation for simulation purpose including mesh generation and assignment of materials properties, sources/load, and boundary condition. Such information is sent to CAE server to solve the associated physical problems. Designers can invoke several numerical analysis processes through CAE server and cooperatively analyze the product performance. After the design task is completed, the task information is updated to the product design workflow system.

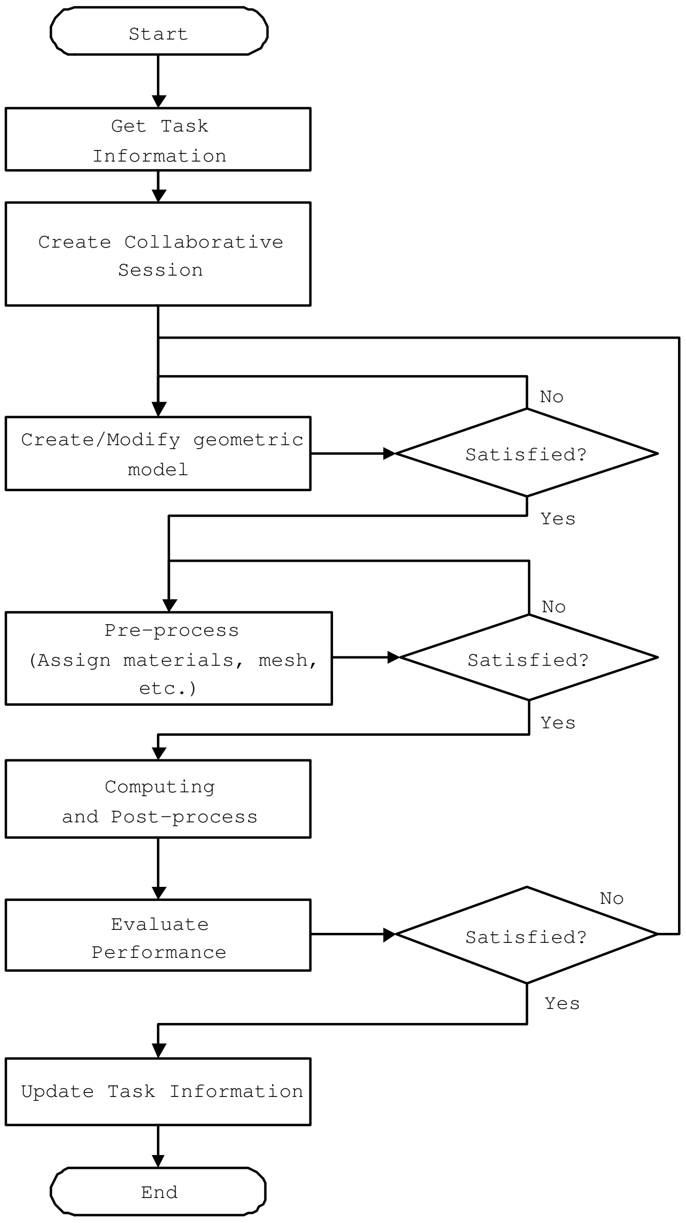

Start
Get Task Information
Create Collaborative Session
Create/Modify geometric model
Satisfied?
No
Yes
Pre-process (Assign materials, mesh, etc.)
Satisfied?
No
Yes
Computing and Post-process
Evaluate Performance
Satisfied?
No
Yes
Update Task Information
End

Figure 3.6: A typical product design flow in the product design system

3.3.1 Presentation Tier

Product design client (Figure 3.7) covers all necessary modeling and analysis facilities needed for product design. It supports designers to work on their own design tasks asynchronously or invoke a session to discuss problems and work cooperatively with other designers. Based on the geometrical modeling kernel, ACIS [58], product design client deals primarily with geometric-based data and describes the initial parts of a product. It uses Hoops Stream Format (HSF) [62] to support data streaming and visualization of product data and computing results. In addition,

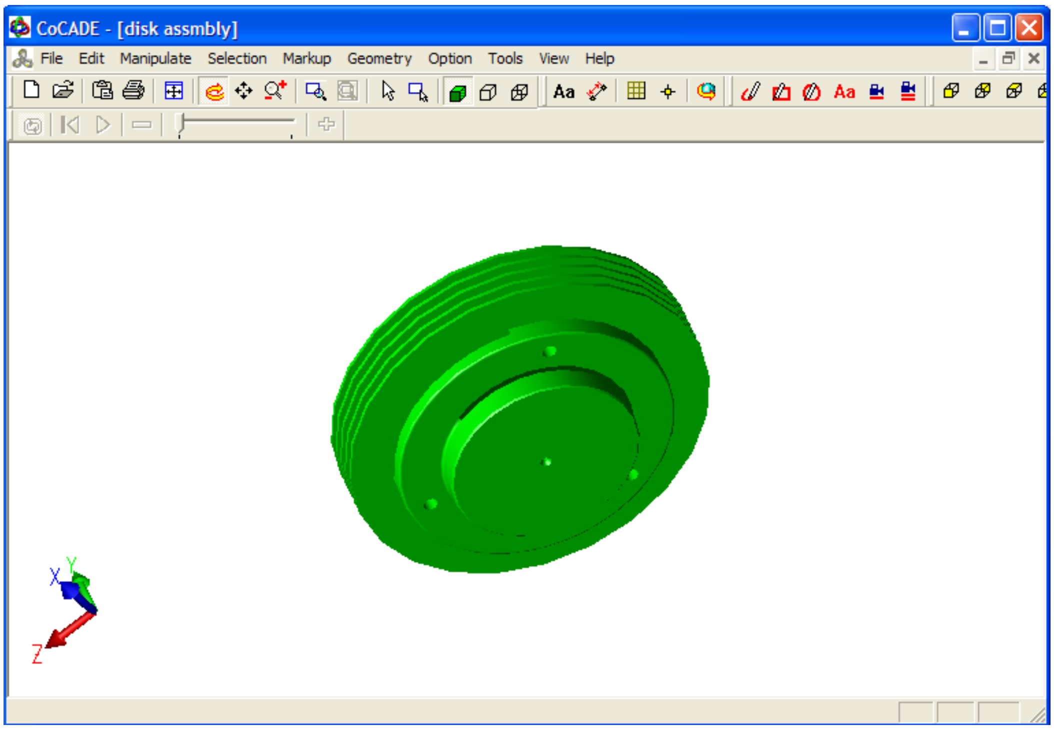

CoCADE - [disk assmbly]
File Edit Manipulate Selection Markup Geometry Option Tools View Help
X
Y
Z

Figure 3.7: Product design client drawing disk assembly

product design client manages other parameters necessary for product performance evaluation, such as material properties, physical loads, boundary conditions and environment conditions. The information flow in product design client is described in Figure 3.8. Three types of information data are generated during the collaborative product design process: operation information, HSF presentation stream, and session state. All the data are transmitted through .NET Remoting channel. And all clients are synchronized during the design process.

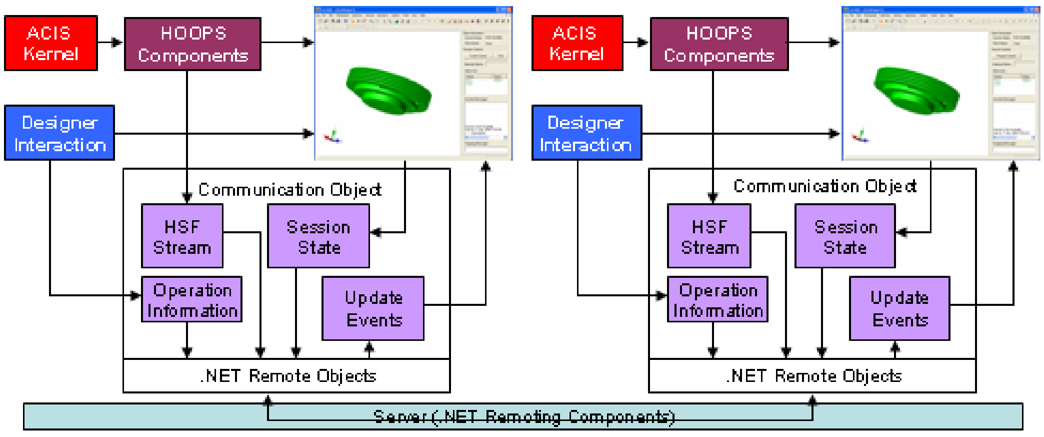


Figure 3.8: Information flow in the product design client

3.3.2 Business Logic Tier

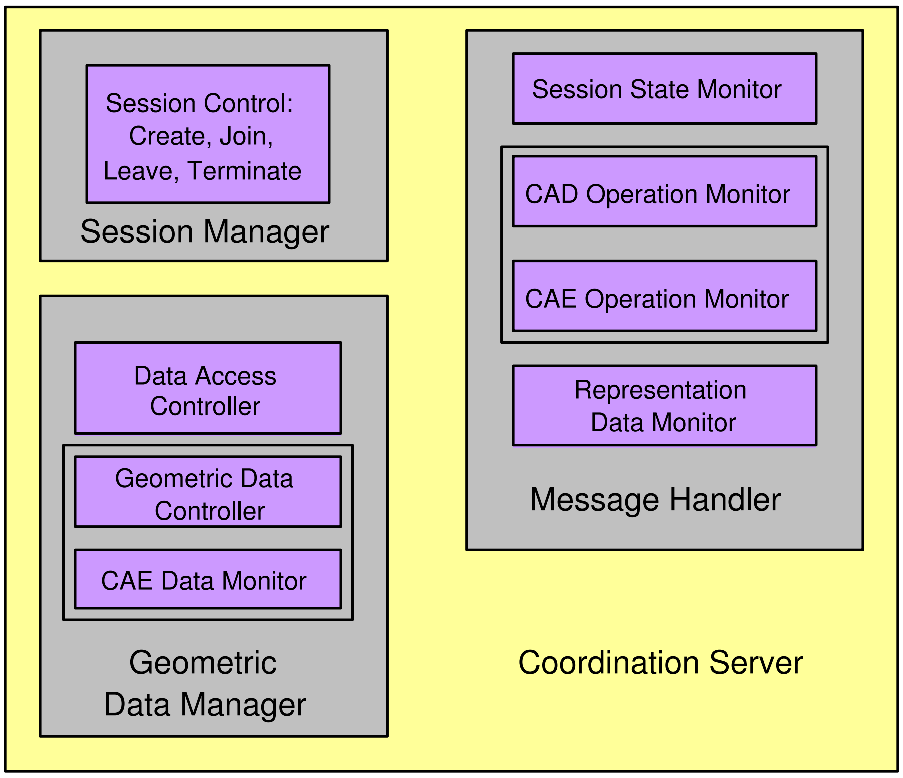


Figure 3.9: Coordination server structure

Business logic tier includes two parts. i.e. Coordination Server and CAE Server. Coordination Server (Figure 3.9) is mainly implemented with three parts: Session Manager, Message Handler and Geometric Data Manager. Session Manager has session management functions including creating session, joining session, leaving session and terminating session. Message Handler handles all the messages needed for th collaborative design and analysis. It consists of Session State Monitor, Operation Monitor, and Representation Data Monitor. Session State Monitor watches changes in the session state, such as join-in of a new client, session termination, etc., and provides the information about the new session state to all clients. Operation Monitor watches new CAD operations and CAE operations. Representation Data Monitor watches new HSF data which is used for visualization. Geometric Data Manager manages product data access. It associates product data access privilege with user management to avoid data divulging and handle data access conflicts to keep data consistency. Geometric Data Controller controls CAD data transmission and storage (in public database at server side). CAE Data Monitor controls CAE data (both simulation results and post-process results) transmission.

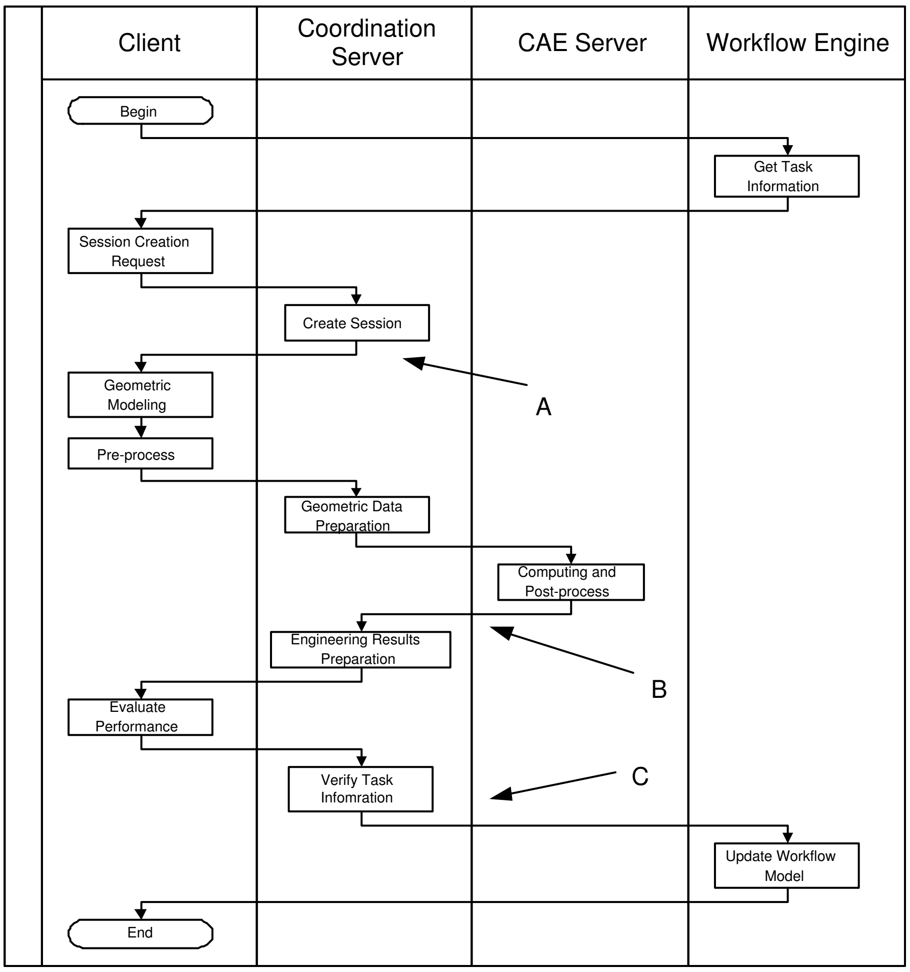


Figure 3.10: Coordination flow for a typical design process

Coordination Server provides all coordination functions for collaborative engineering design and analysis. Figure 3.10 shows the coordination flow for a typical product design process (Figure 3.6). First, it is implemented with network protocol drivers (TCP/IP and HTTP) to support reliable connection over Intranet/Internet, provide .NET remote components to manage sessions and message circulation for product design clients (A in Figure 3.10). Second, it handles coordinates with CAE server for real-time product performance evaluation (B in Figure 3.10). Finally, Coordination server can verify product design sessions and update task information to workflow engine using XML messages (C in Figure 3.10). That is, when a collaborative session finishes, the corresponding updated XML message is sent from the coordination server to the workflow engine, and then the work flow runs to the next task. The communication framework is shown in Figure 3.11.

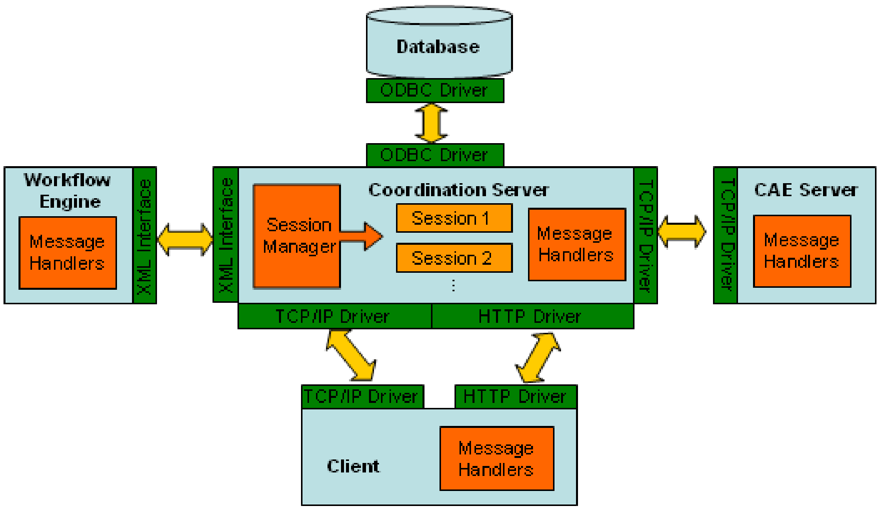
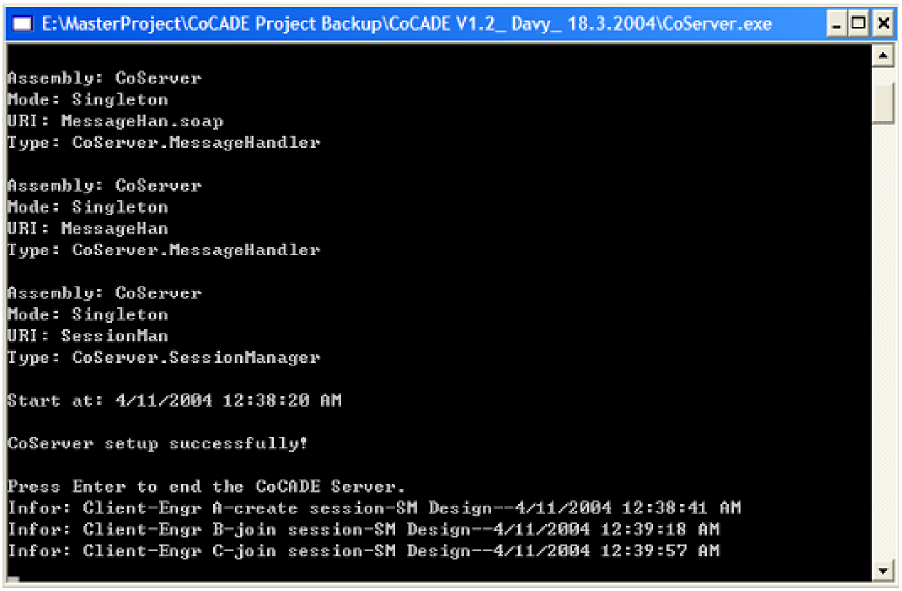


Figure 3.11: Communication framework in coordination process

During the product design process, the run-time information of collaborative session can be monitored at the coordination server (Figure 3.12).



E:\MasterProject\CoCADE Project Backup\CoCADE V1.2_ Davy_ 18.3.2004\CoServer.exe

```
Assembly: CoServer
Mode: Singleton
URI: MessageHan.soap
Type: CoServer.MessageHandler

Assembly: CoServer
Mode: Singleton
URI: MessageHan
Type: CoServer.MessageHandler

Assembly: CoServer
Mode: Singleton
URI: SessionMan
Type: CoServer.SessionManager

Start at: 4/11/2004 12:38:20 AM

CoServer setup successfully!

Press Enter to end the CoCADE Server.
Infor: Client-Engr A-create session-SM Design--4/11/2004 12:38:41 AM
Infor: Client-Engr B-join session-SM Design--4/11/2004 12:39:18 AM
Infor: Client-Engr C-join session-SM Design--4/11/2004 12:39:57 AM
```

Figure 3.12: Coordination server application

Based on .NET technology, coordination server provides coordination for multiple product design clients. Table 3.1 shows the configuration of coordination server. Two channels, i.e. TCP/IP channel and HTTP channel, are opened to

Table 3.1: Configuration of coordination server

Channel	Wellknown Mode	Component	Object Uri
HTTP	Singleton	Session Manager	SessionMan.soap
HTTP	Singleton	Message Handler	MessageHan.soap
TCP/IP	Singleton	Session Manager	SessionMan
TCP/IP	Singleton	Message Handler	MessageHan

the product design client. Two components, i.e. the session manager and message handler, are implemented to provide coordination services.

The CAE Server provides all necessary functions for product evaluation, including primarily simulation tools and perhaps heavy duty mesh generation and post-process tools. It receives the instructions/parameters from the coordination server, and executes corresponding process. Meanwhile, it manages engineering results generated by the CAE tools and stores the computational results and necessary information to the data tier.

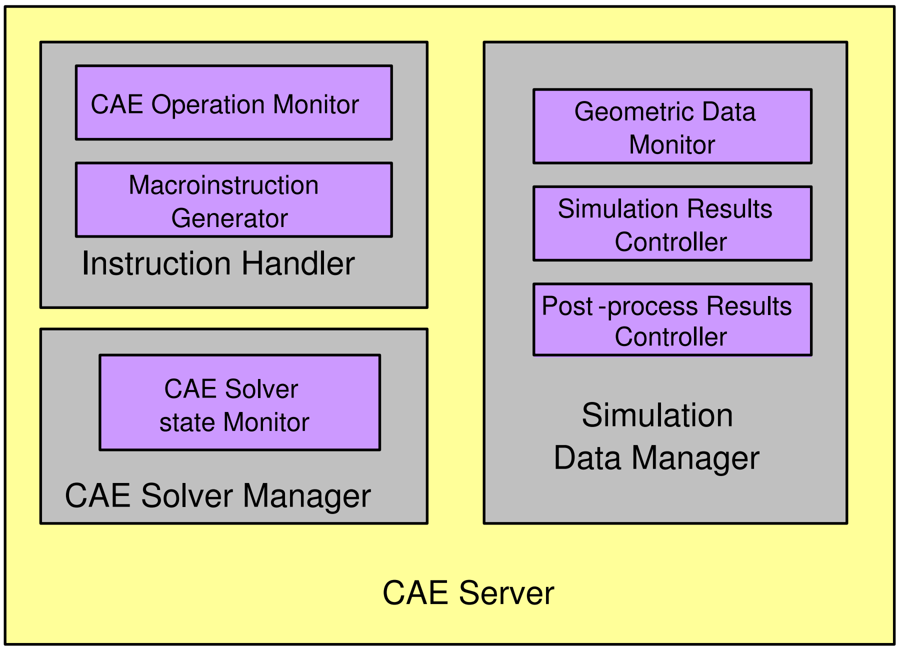


Figure 3.13: CAE server structure

As shown in Figure 3.13, CAE Server was implemented with Instruction Handler, Simulation Data Manager and CAE Solver Manager. Instruction Handler watches incoming CAE operations from Coordination Server and generate corresponding macroinstruction stream to activate CAE solver to perform computing tasks. Simulation Data Manager manages engineering data that are generated during computing process. CAE Solver Manager controls the different CAE solvers to perform computing tasks.

3.3.3 Data Tier

The data tier provides data management for the entire product development process. It deals with user information, product design process information, CAE tools information, pre-/post-process parameters, product geometric model information, simulation/computing results information and development history. As shown in Figure 3.14, the product information can be grouped into the following views:

- **Product Design:** This view covers most of the aspects needed for CAD functions in product design client. It deals with geometry-based data and the initial parts of the product are described.
- **Product Design Process:** This view records the product design workflow. The product design processes are defined. The task specifications and status are stored.
- **Simulation and Analysis:** The simulation and analysis view describes the additional data needed for the analysis and the simulation domain. In addition to the test data, the results are stored to provide the basis for comparison of different versions of the product design.

The product database should be a group database, accessible by all the users involved in the design of the product. Checkin and checkout mechanism should be used to keep the operations in a cooperative, controlled, and predictable manner.

- **Check-ins:** Check-ins are useful for moving objects from a personal database to the group database. One could either check in objects which were earlier

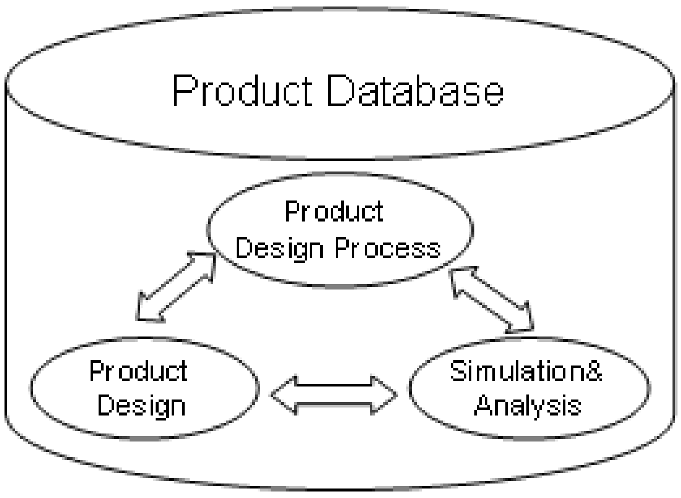


Figure 3.14: Product database

checked out from a group database or check in the ones created in the personal database.

- **Check-outs:** Check-outs are useful when the user wants to work on a particular object for a long period of time with a minimum of network traffic.

If the designer makes changes to an object and then decides to commit the changes to the database, then the new version has to be created. Just before checkout or checkin, the product design client should check if the object is versioned or not. If the object is not versioned, it should be converted into a versioned object. This will ensure that every time an object is checked in, a new version is created. Periodically each product design client should group the latest state of all the objects used by the client and create a new version of this group of objects. This will be useful in the later design stage, when the individual components of the product will be assembled together to form a product model.

3.4 Architectural Overview of Collaborative Session

Traditional session definition [7] is lack of a systematic and comprehensive functional description for collaborative design. Thus, the definition of session for collaborative product design has been extended as:

The process in which multi-discipline designers, who may be from geographically dispersed locations, work together to design product or analyze engineering results, synchronously or asynchronously, with the help of collaboration tools.

The difference between an asynchronous collaborative session and a synchronous collaborative session is that, in an asynchronous collaborative session, collaborators can carry out different tasks in different workplaces asynchronously and cooperatively; while in a synchronous collaborative session, collaborators carry out the same task in the same workplace.

Apart from the description of session functions, it is necessary to describe the supporting infrastructure for an effective collaborative session. The main issues related to the collaborative session management may include: the relation between session and design task, the manner of session coordination, and the synchronization requirements in the collaboration process. The coordination and synchronization issues in synchronous collaborative session are much more complex than that in asynchronous collaborative session, as more real-time interactions are needed for an effective synchronous collaboration process. Figure 3.15 presents the architectural structure of collaborative session.

From Figure 3.15, it can be seen that the primitive objective of a collaborative session is to realize co-design and co-analysis with the help of collaboration tools, such as view, highlight, mark-up and annotation. The messaging function helps collaborators to communicate ideas and discuss on the geometric model or engineering results. The logging function can record the whole design process that

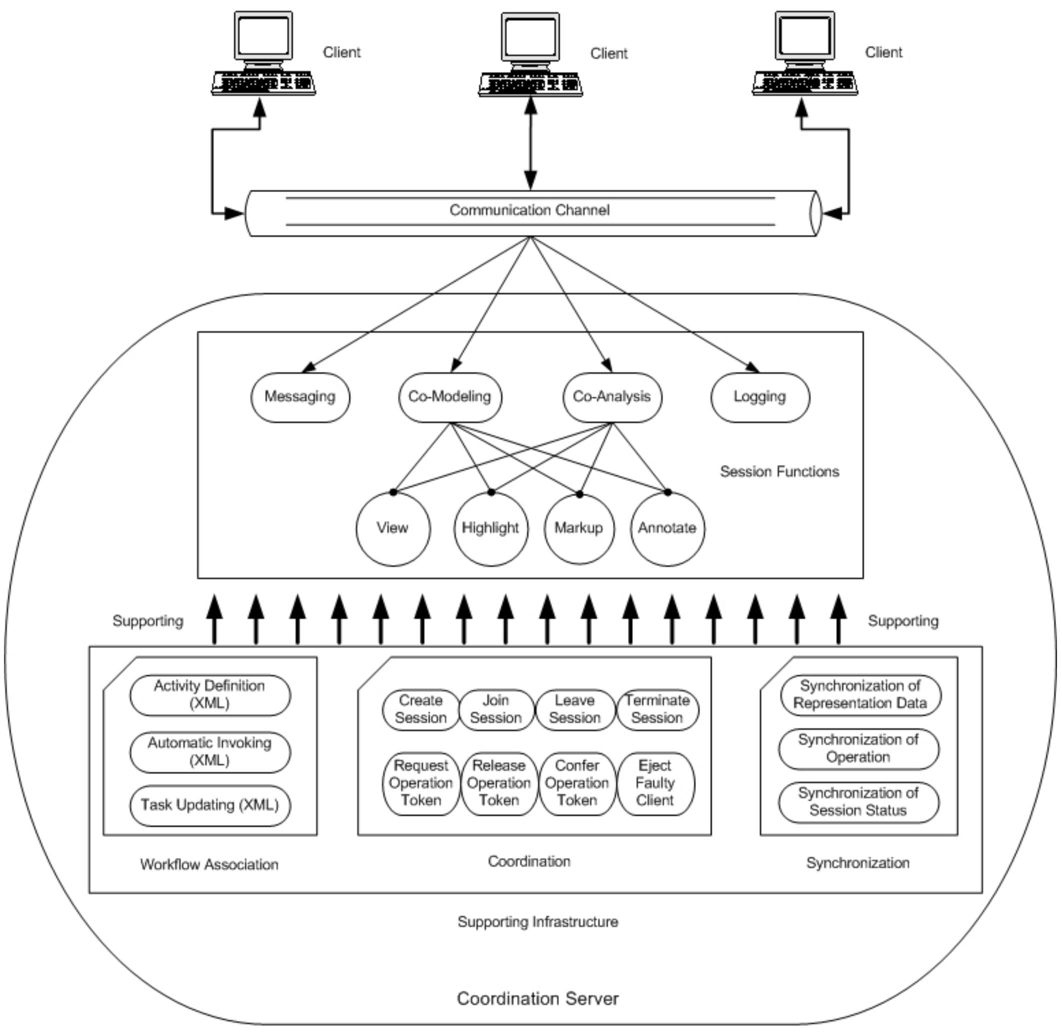


Figure 3.15: Architectural structure of collaborative session

is carried out in a collaborative session.

In order to effectively support above functions, the following important issues should be addressed.

- **Workflow Association**

To provide effective support for a collaborative design activity, it is necessary to establish the link between the collaborative session and the specific design task. Every design activity should be pre-defined and associated with the product design workflow. Collaborative sessions can be automatically defined, this may include definition of resource requirements, dependency constrains, etc. to facilitate design activities. Also, upon terminating a collaborative session, the corresponding task information should be updated.

In this research work, a product design workflow system was incorporated into the engineering design and analysis environment to support collaborative sessions using a workflow-driven mechanism.

- **Coordination**

Coordination is the process that allows individuals to work together, which involves communication between the participants. It includes the mechanism of session establishment. As shown in Figure 3.15, the key functions for session establishment are: creating session, joining session, leaving session, and terminating session. Operation token management is also needed for synchronous collaborative session to avoid operation conflicts. The key functions for operation token management are: request token, release token, confer token, and eject faulty client (Figure 3.15).

Coordination is important for a collaborative session, especially for a synchronous session in which collaborators join online and carry out the same design task in real time. In this thesis, the coordination mechanism in a synchronous collaborative session is investigated and discussed in detail.

- **Synchronization**

Synchronization is another important issue for collaborative session management. It is particularly vital for synchronous collaboration process in which the representation data, operation information and session status should be synchronized in order to support an effective and efficient design activity. Figure 3.15 shows the synchronization requirements for synchronous collaborative session. A new synchronization scheme for synchronous collaborative session management is proposed in this research work.

3.5 Summary

In this chapter, the architecture of a distributed collaborative CAD/CAE framework, CoCADE, has bee proposed. A product design workflow management system is integrated in the framework to manage all collaborative activities. Based on .NET remoting technology, the product design system is implemented using a 3-tiered Client/Server architecture to support collaborative product design. The architectural structure of collaborative session has been presented and the critical issues that affect collaborative session have been discussed. The framework provides a reliable solution for collaborative engineering design and analysis in a distributed environment.

Chapter 4

Collaborative Session Management

The CoCADE framework can provide reliable support to CAD and CAE sessions participated by designers from geographically dispersed locations. However, in such a collaborative design and analysis environment integrated with CAE, collaborative session management becomes more challenging.

In this chapter, collaborative product design process using CoCADE framework is investigated. Based on the Unified Modeling Language (UML) model, a workflow-driven mechanism has been proposed to organize the collaborative sessions. The coordination and synchronization issues that significantly affect collaboration process are discussed in detail, and corresponding solutions have been proposed.

4.1 Organization of Collaborative Sessions

Engineering product design is viewed as a systematic and iterative transformation from abstract needs to concrete and detailed artifacts [63, 64]. It is typically described as a process, i.e., the product design process. Gero and McNeill [65] have shown that product design can be seen as a series of discrete activities which are carried out at different product design stages.

Collaborative product design process can be treated as a specialized kind of business process, in which documents, information, and tasks are “passed” from one “stage” to another according to a set of rules. The design tasks can be carried out in parallel or sequence. Collaborators work together for moments, then divide up and go in their separate ways [66]. Communication and coordination between relevant tasks are required for effective product design. Overlapping and cross-functional cooperation are essential in the approaches of a collaborative design process [67, 68].

In this section, UML approach, which consists of use case and activity diagrams, was adopted to model the logical perspectives of collaborative product design process in the proposed framework. Based on the UML model, a workflow-driven mechanism is adopted to manage collaborative sessions that facilitate design activities in a collaborative product design process.

4.1.1 Introduction to UML

In order to eliminate the difference between the business description and the software specification, unearthing common language understood by users and developers is imperative. Each symbol and semantic within the language must be defined clearly and intuitively for users. UML is a well-defined and standard modeling language. UML consists of use case, sequence, collaboration, class, object, state, activity, component, and deployment diagrams [69]. A system could be modeled via these diagrams from various aspects, such as structural, behavior, implementation, and environment views.

Use case diagram is useful to represent goals, responsibility, functionality, and boundary intuitively for a business process. It also expresses static interactions between business processes and their external objects. When notations of use case diagram maps into workflow mechanism, use case notations stand for sub-processes of a business process, and actor notations stand for participants [70]. Therefore, based on the internal functions of a business process, each use case notation describes a sub-process, which composes the whole business process. Each use case also can be further detailed in another use case diagram. An actor of use case diagram may be a user, an invoked application, a database, or a legacy system.

Even though use case diagram represents business processes, it cannot show the order of each use case instance and dynamic behavior. Within the UML model elements, both sequence diagram and activity diagram support to describe the dynamic behavior of use cases. Whereas sequence diagram emphasizes the flow of control from object to object, activity diagram emphasizes the flow of control from activity to activity [71]. In contrast to sequence diagram, activity diagram is very useful in modeling the process definition of the workflow and in describing the behavior that contains a lot of parallel processing. This is essential for a collaborative product design process.

In the following, UML approach is used to specify process definition. Use case diagram is adopted to express the specification of system functionality, goals, responsibility and iterations. Then activity diagram is adopted to model business logical steps and dynamic behavior derived from previous use case diagram. Finally, the workflow-driven mechanism is described based on the UML model.

4.1.2 Collaborative Product Design Process

If the tasks of product development process are performed separately, the high level of interdependence may lead to error and critical situations [72]. In order to capture the context of a collaborative product design process in proposed framework, object relationships are presented with use case diagram. Figure 4.1 shows a use case diagram for a typical product design process. The diagram contains four use cases and five actors.

In Figure 4.1, planner has the role to plan the whole product development process. Designers follow the pre-defined workflow and perform product design synchronously or asynchronously. Product design workflow system accepts task specifications and generates product design workflow. Coordination system acts as a coordinator among product design, evaluation and simulation. CAE solver focuses on the simulation of product model.

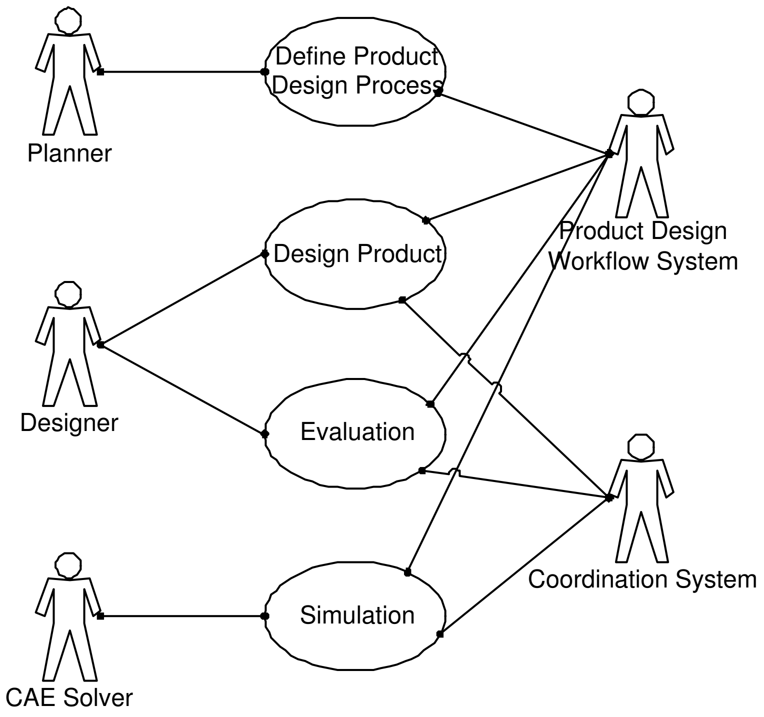


Figure 4.1: Use case diagram for a product design flow

Collaborative design process consists of a series of discrete, sequent or parallel, activities which have interdependencies at certain stages. Collaborative sessions are established to facilitate these activities. Figure 4.2 is the activity diagram for hard disk spindle motor design flow. It consists of asynchronous collaborative sessions (e.g. A and B in Figure 4.2) and synchronous collaborative sessions (e.g. C and D in Figure 4.2).

When collaborating asynchronously, designers can carry out different design tasks. As shown in Figure 4.2, the stator and rotor design tasks in stage A may be interdependent, as the stator and rotor are to be assembled to spindle motor. The dependable computing tasks in stage B may be highly interdependent.

Asynchronous collaborative sessions are established to facilitate these dependable tasks.

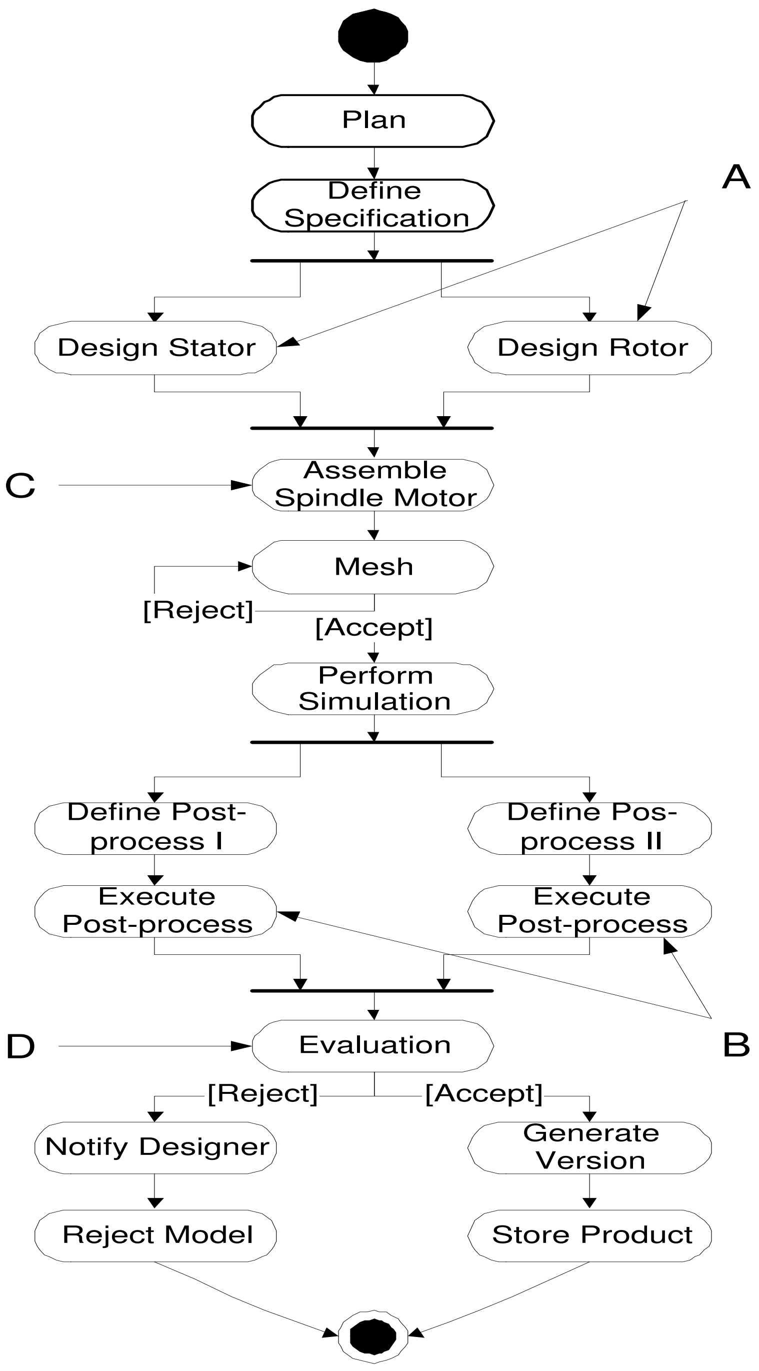


Figure 4.2: Activity diagram for spindle motor design and analysis flow

When collaborating synchronously, engineers can carry out the same task cooperatively in real time (e.g. C or D in Figure 4.2). Synchronous collaborative sessions are established to faciliate such activities. In synchronous collaborative session, designers work intensely with one another, observing and understanding each other's intentions. Each participant contributes what they can in different fields of expertise at moments when they have the knowledge appropriate to the situation.

4.1.3 Workflow-driven Collaborative Session Management

Due to the complexity and interdependency of product design process, there may be a lack of common understanding among the participants. Thus, efficient arrangement of the workflow activities can greatly enhance the performance of the whole design process. During the execution of workflow model, changes will have to be made to suit new environments. Hence, a dynamic characteristic is imported to the workflow which provides the functionalities that activities can be added or dropped and collaborative sessions can be defined automatically to facilitate new activities, the activities' profile can also be altered even when the workflow process is running. These are achieved by the flexible definition of the activities and configuration of the workflow engine. Figure 4.3 shows an example of workflow model in product simulation stage.

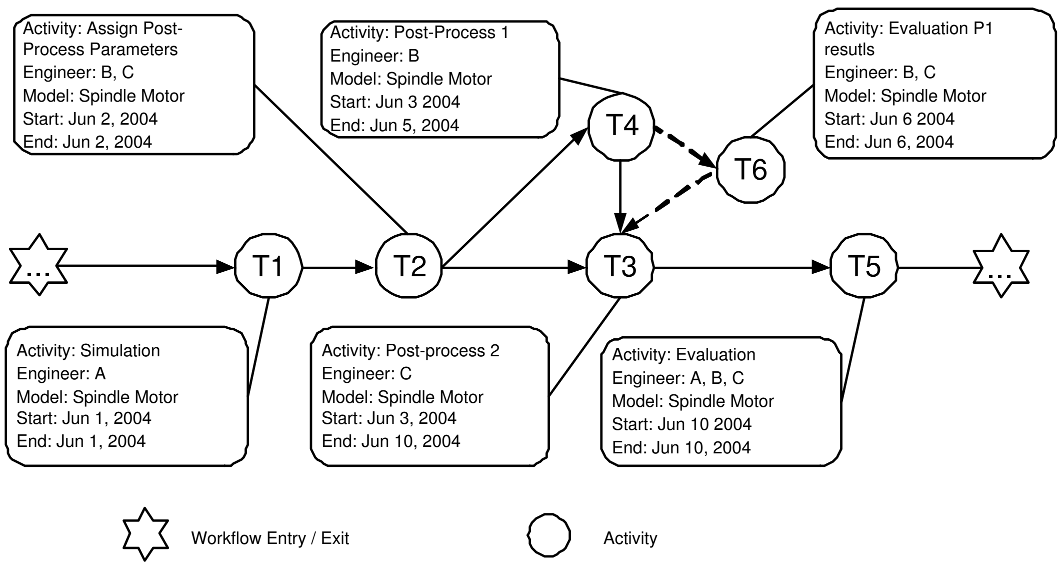


Figure 4.3: An example of workflow model in product simulation stage

In Figure 4.3, every task consists of related information, such as activity, engineer, model, time requirements etc. T_3 and T_4 are dependable computing tasks: T_3 needs the computing results of T_4 during its computing process before it proceeds to T_5. In the initial workflow model, T_4 is expected to finish before receiving the data request (on Jun 5 onwards) from T_3. The old workflow route is $T_4 \Rightarrow T_3$. T_3 and T_4 are performed in asynchronous sessions. However, due to requirements change of product model, it needs to evaluate the results generated by T_3 before sending them to T_4. In such case, a new task T_6 is dynamically defined in the workflow model while T_4 and T_3 are executing. Before T_6 is inserted between T_3 and T_4, operations such as checking states of T_4 and T_3 are performed by workflow engine. After passing verification, T_6 is added and waits for execution, and then a synchronous collaborative session is defined by workflow system to carry out T_6. The new workflow route becomes $T_4 \Rightarrow T_6 \Rightarrow T_3$. If the new task cannot pass the verification of workflow engine, workflow system will inform designers through coordination server to manually modify the workflow model to facilitate the new task.

During the execution of the product design process, collaborative sessions are managed by workflow model in which all task specifications are defined. When task attributes are changed or the connectors between different tasks are redirected, synchronous or asynchronous collaborative sessions are defined to facilitate these changes. Eventually, the product design workflow model can improve the flexibility and changeability of product development by effectively organizing collaborative sessions.

4.2 Synchronous Collaborative Session Management

Communication and coordination are essential for an effective collaboration process, especially for synchronous collaborative sessions when collaborators need to work on the same product model simultaneously. This section will investigate the primary aspects of synchronous collaborative session. Coordination and synchronization issues that significantly affect collaboration process are discussed in detail, and corresponding solutions have been proposed.

4.2.1 Data Security and Consistency

The security issues in distributed collaborative engineering system can be broken into three categories: Client Security, Transmission Security, and Collaboration Security. Standard solutions can be used for Client and Transmission Security, such as public-key encryption [73]. Collaboration Security should be considered in synchronous collaborative sessions. The main challenge for Collaboration Security is the fact that the collaborator must divulge information to the online product design workplace when collaborating in synchronous collaborative session, yet the collaborator needs assurances that this does not let others learn the details of their design.

A distributed collaborative engineering design and analysis system has to ensure every client has the same view of data including product data and engineering data. The simplest way to realize this is to store data only once on the server and to redirect any access to these data via RPC. However, for a distributed CAD/CAE framework which needs large scale engineering data exchange, RPC solutions are not adequate [74]. Another way of sharing data is to provide each client with a replica of the data and to ensure the consistency between each two replicas. Replication process consists in copying on the client the data that the remote program needs among those stored on the server, and in ensuring the consistency at each modification realized on a data or on one of its replicas.

In CoCADE system, a public database at server side is deployed to store all versions of product data. Designer can check out the product data to his local database for asynchronous/synchronous collaborative design or check in the latest product data.

The measure is used to manage product data in a synchronous collaborative session is that only one client owns the initial data for editing or analysis. Other clients only have the necessary data for visualization. Through doing so, it is easy to keep data consistency compared with the system in which each client has one copy of data. It uses RPC to realize efficient information exchange over network. This measure makes conflict control easy in the whole design session. It can also be utilized as a measure to secure the confidential data that only belongs to one company or team.

4.2.2 Coordination Mechanism

The synchronous collaborative session is usually group-based. The initiator who creates the collaborative session plays the leading role in the collaborative design process. Others are collaborators who join the session. Accordingly, Centralized Coordination Mechanism (CCM), including centralized session management and centralized token circulation management, has been developed for the CoCADE system.

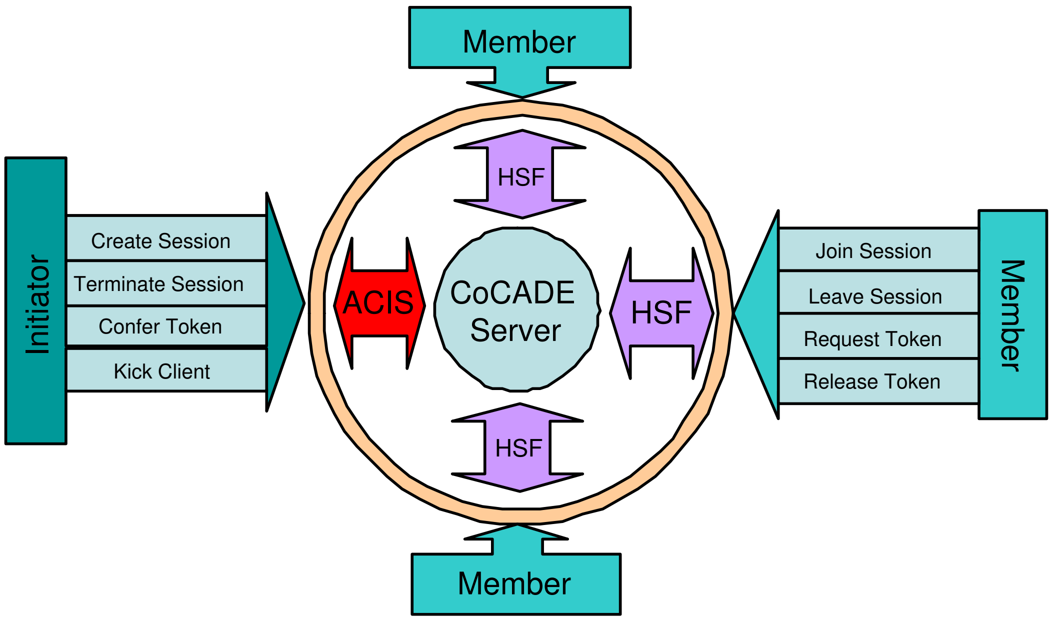


Figure 4.4: Centralized coordination mechanism (CCM)

As shown in Figure 4.4, CCM has two types of clients, Initiator who creates the collaborative session, and Member who joins the collaborative session. Only one client holds the original data that are downloaded from public database or created in the design process, as designated by the initiator. To get the best performance, CoCADE transfers original data only between the server and the data holder (designated by the initiator). The data transferred between two clients is HSF data which is only for visualization. This mechanism helps to reduce the unnecessary data transition and improve network performance.

A token ring protocol is deployed in CCM. Only the designer who owns the control token, named Token Holder, can make changes to the product model. However, the initiator has the full control of operation token. He can not only confer the control token to a requesting client, but also take back the operation token from a "dead" client (e.g., due to network congestion).

Members should obtain confirmation from the initiator and current token holder before he obtains the operation token. If multiple members request the token, the initiator has the right to select the next token holder to avoid conflicts.

Figure 4.5 shows the messaging structure of CCM. The messages during session establishment process and collaboration process are managed by Coordination Server. A typical coordination flow under CCM can be described as follows.

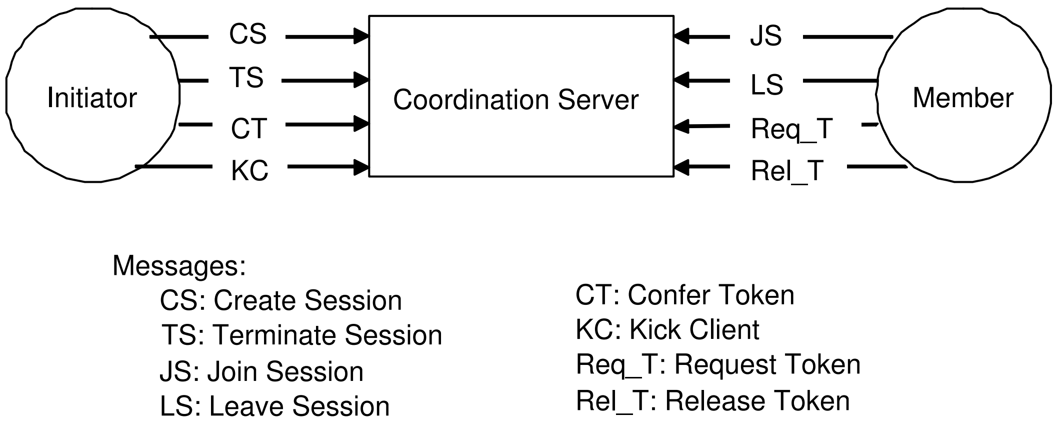


Figure 4.5: Messaging structure of CCM

Session Establishment: Initiator sends Create Session message to Coordination Server. Coordination Server creates the new synchronous collaborative session.

The operation token initially belongs to the initiator. Member sends Join Session message to Coordination Server in order to join the session. When a new member joins, other members that are already in the session will be notified.

Collaboration: Members who want to operate on product model send Request Token message to Coordination Server. Coordination Server informs the Initiator and Token Holder, then waiting for their response. Token Holder sends Release Token message to Coordination Server when his operation is complete. Then Initiator selects next token holder and sends Confer Token message to Coordination Server. Upon receiving Confer Token message from Initiator, Coordination Server notified the selected member.

In such way, the token circulates among designers until the collaborative design work finishes. Using CCM, the collaborative design is kept in a controlled, cooperative and efficient manner.

4.2.3 Synchronization Scheme

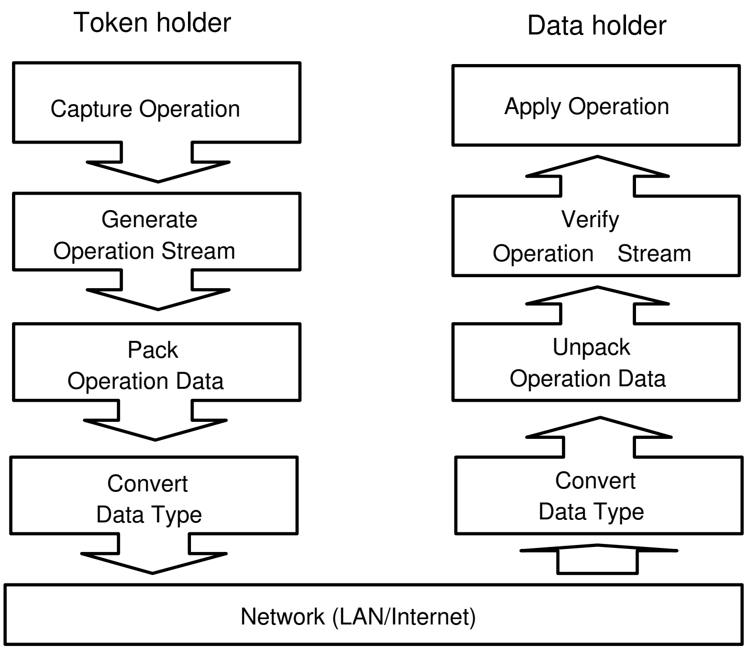


Figure 4.6: Generation of operation information

In a synchronous collaborative session, synchronization is one of the most critical issues that will affect effectiveness and efficiency of collaboration process. It includes the synchronization of operation, initial representation, and session status.

Operation synchronization covers the dynamic synchronization of operation information. This means that when token holder is operating (creating models, changing camera position etc.), the operation information is captured and streamed, then sent to other clients. The process for generating operation information is shown in Figure 4.6

The representation data includes the necessary data for the representation of geometric model, meshing, simulation, and computing results. When new data are loaded into the workplace, the corresponding HSF data that is only for visualization is generated and sent to other clients. Initial representation synchronization ensures that all clients share the same view of the newly loaded data. The generation of representation data is shown in Figure 4.7.

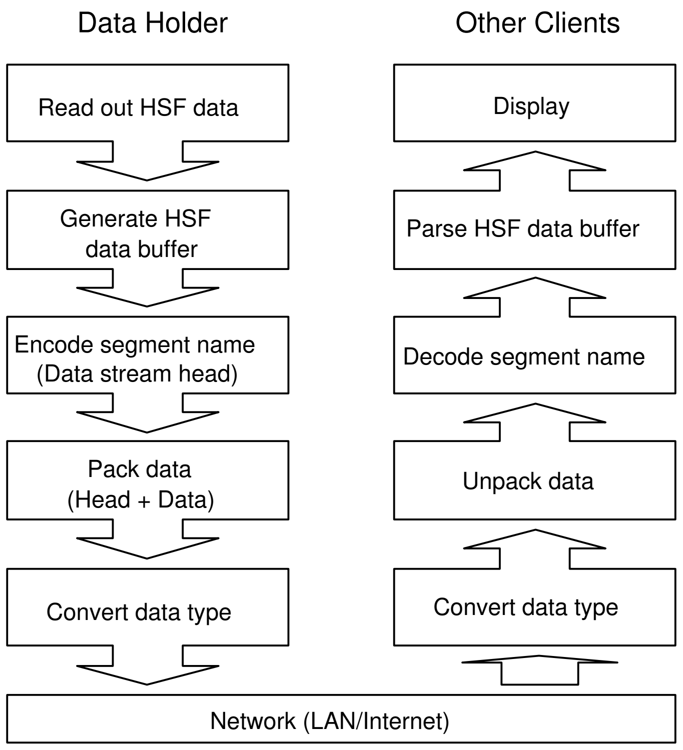


Figure 4.7: Generation of representation data

Session status synchronization means every client should know other clients' status. That is when a client requests the control token, other clients should be notified.

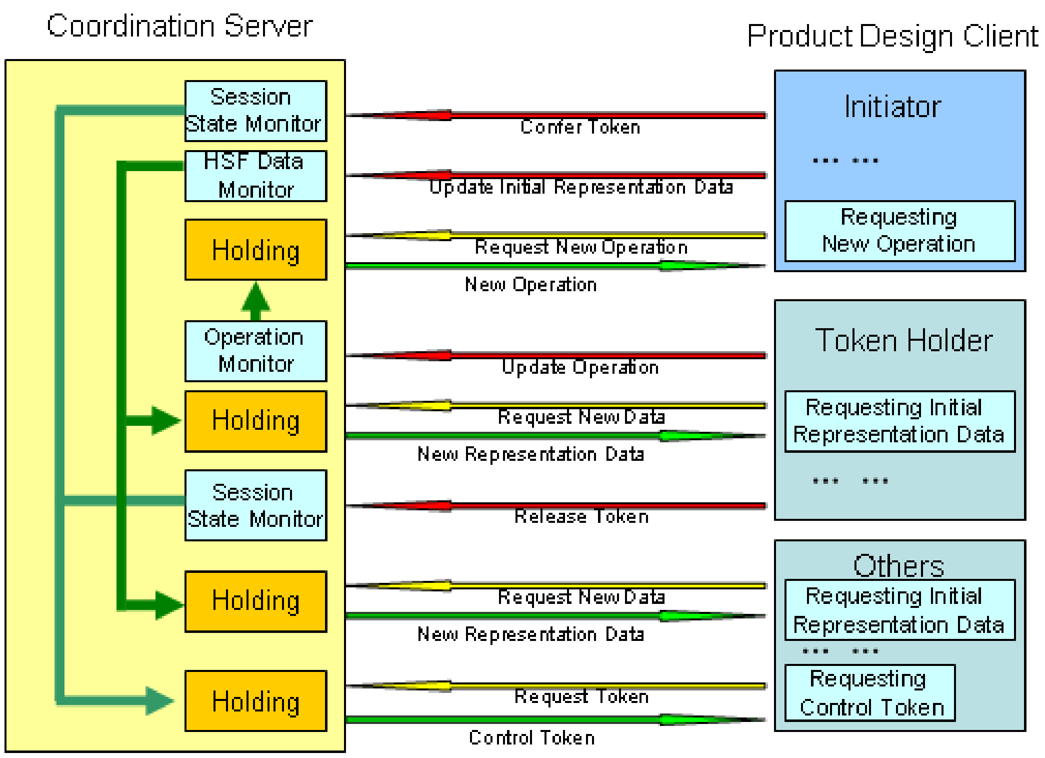


Figure 4.8: Multi-thread request response (MTRR) scheme

The Multi-thread Request Response (MTRR) scheme (Figure 4.8) has been implemented to realize concurrent collaboration in CoCADE system. Upon joining the collaborative session, the client initiates three threads to request updated information including initial representation, operation, session status from Coordination Server.

Figure 4.8 shows an example. The initiator requests the new operation. Other clients request the initial representation data. The clients without the control token might request control token. If the information (operation, initial representation, token etc) is not available, the request will be hung up at Coordination Server.

Three monitors including Session State Monitor, Operation Monitor and HSF Data Monitor are implemented in Coordination Server. They are responsible for monitoring the status of session (e.g. token state), operation and initial representation data respectively.

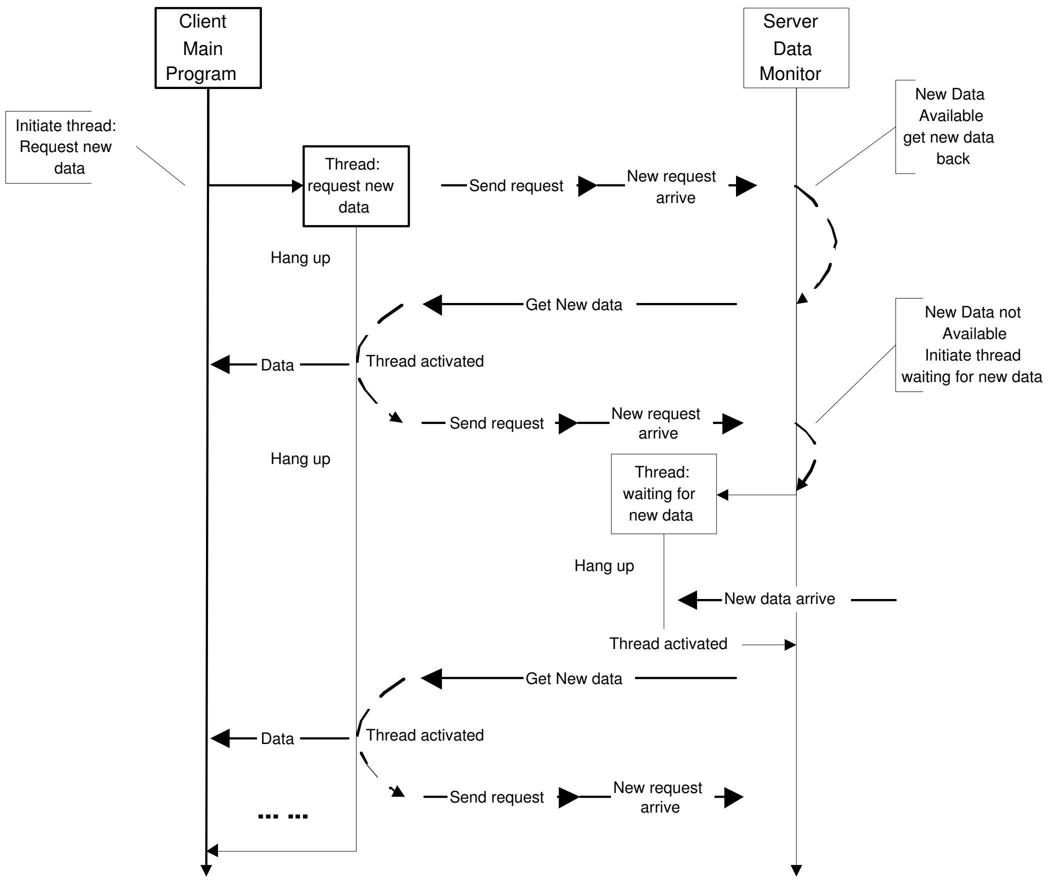


Figure 4.9: Realization mechanism of MTRR scheme

Figure 4.9 shows the realization mechanism of the MTRR scheme. Client main program initiates a thread to request new data. Upon sending data request to coordination server, the thread hangs up and is waiting for the response from server. Server data monitor receives the data request from client and checks whether the requested data is available. If the data is available, the server immediately sends the data along with the return results of the request. Upon receiving data from the server, the client thread is activated. It forwards the new data to client main program. It then sends a new request to server and hangs up.

If the data is not available at the server side, the server has to wait for the arrival of the new data from other clients (data holder in this example). Therefore, to hold the request for the new data, the server initiates a new thread to hold the request. That is, a thread is initiated after receiving the request and hangs up to block the execution of the corresponding server program - the function that deals with data requests from the client. Once receiving new data from the data holder, the hanging thread is activated, then the corresponding server program proceeds to get the new data and assigns them to the return results of the request.

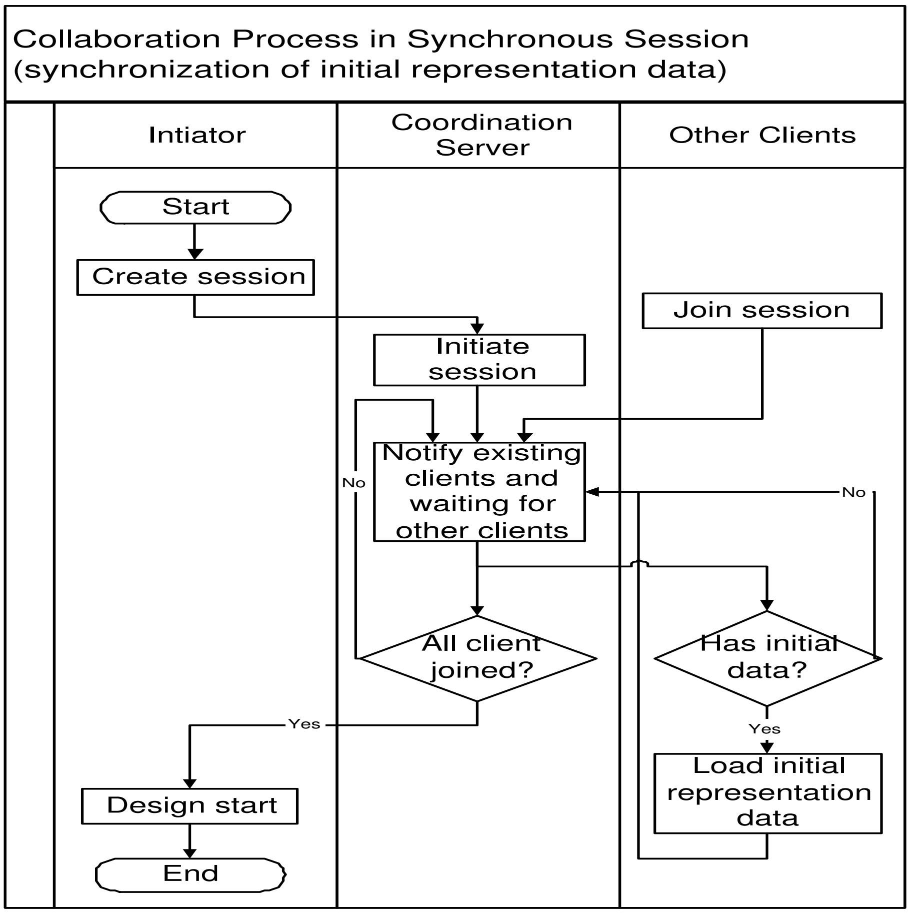

Collaboration Process in Synchronous Session (synchronization of initial representation data)
Intiator
Coordination Server
Other Clients
Start
Create session
Join session
Initiate session
Notify existing clients and waiting for other clients
No
No
All client joined?
Has initial data?
Yes
Yes
Load initial representation data
Design start
End

Figure 4.10: Synchronization of initial representation data

By such way, MTRR scheme can ensure each client to receive updated information effectively. It can achieve following benefits:

Robustness: The clients do not need to hold and expose a public IP. In other words, the server does not need to know every details of client. This is very important for designers, who resided in a different network domain of different companies, to collaborate over Internet.

Efficiency: The server response time is reduced especially when the required information is not available. Client needs not to repeat sending request to server. It can get new information immediately after new data arrives on the server side.

Effectiveness: Every request has an effective return result since the request will be held if it cannot obtain desired information. Hence it can realize instant response with minimum requests and improve network utilization.

The communication framework that supports MTRR scheme and the operation latency using MTRR scheme will be discussed in the succeeding sections. The processes using MTRR scheme to synchronize initial representation data, operation and session status are described below.

When data holder (the initiator in this example) loads product geometric data (or computing results, etc.) to workplace, the corresponding HSF data is generated and transmitted to Coordination Server. The initial data monitor will activate the sleeping requests for initial representation data. Then other clients will receive the initial representation data back to update their visualizations. The synchronization flow of initial representation data is illustrated in Figure 4.10.

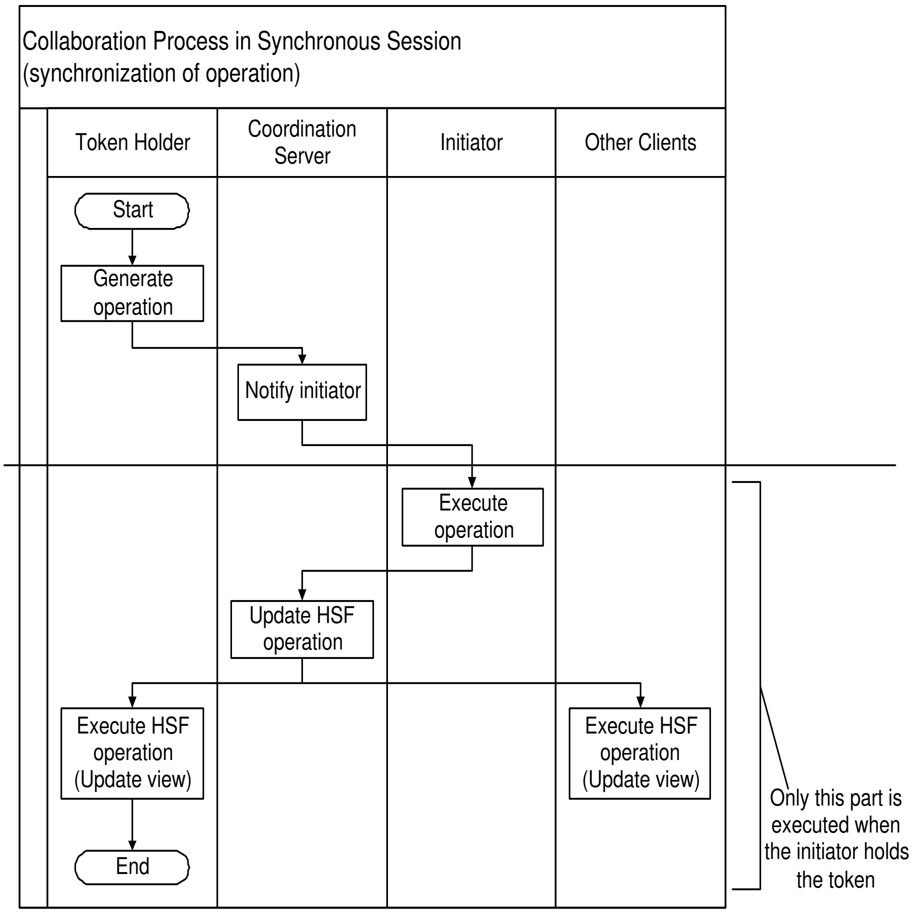

Collaboration Process in Synchronous Session
(synchronization of operation)
Token Holder
Coordination Server
Initiator
Other Clients
Start
Generate operation
Notify initiator
Execute operation
Update HSF operation
Execute HSF operation (Update view)
Execute HSF operation (Update view)
End
Only this part is executed when the initiator holds the token

Figure 4.11: Synchronization of operation

When the token holder executes an operation, the operation information is sent to Coordination Server. The operation monitor activates the sleeping requests for new operation. Operation is executed at data holder (the initiator in this example) and corresponding HSF operation information is broadcast to other clients. Then other clients receive the operation information and execute on the basis of HSF data that is only for visualization. The synchronization flow of operation is shown in Figure 4.11.

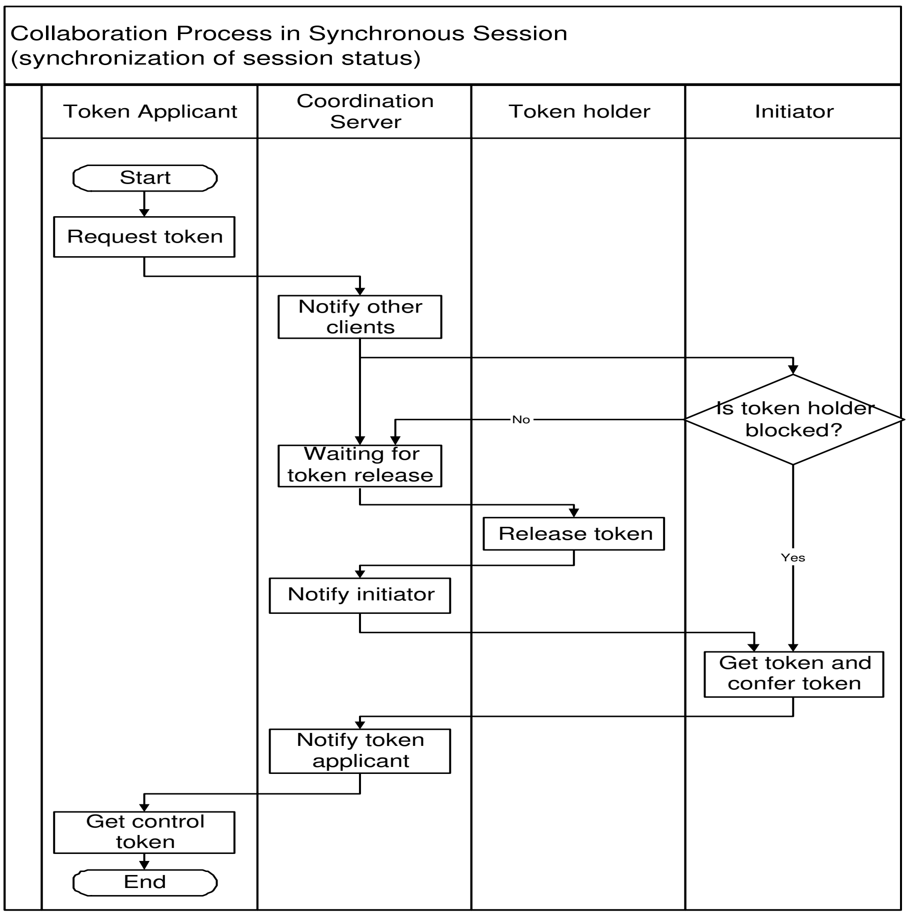


Figure 4.12: Synchronization of session status

When session status changes, such as requesting token, every client should receive the same information. Figure 4.12 shows the synchronization flow of a token circulation process. When a client requests the control token from token holder, he should obtain confirmation from initiator and current token holder before he obtains the operation token. And the initiator has the right to select the next token holder to avoid conflicts.

4.2.4 Communication Framework

The MTRR scheme is developed based on .NET remoting technology. Figure 4.13 shows the .NET remoting architecture. The initial representation data (HSF stream), operation and session status are synchronized between two clients. The update events include the new representation data, new operation or new session state.

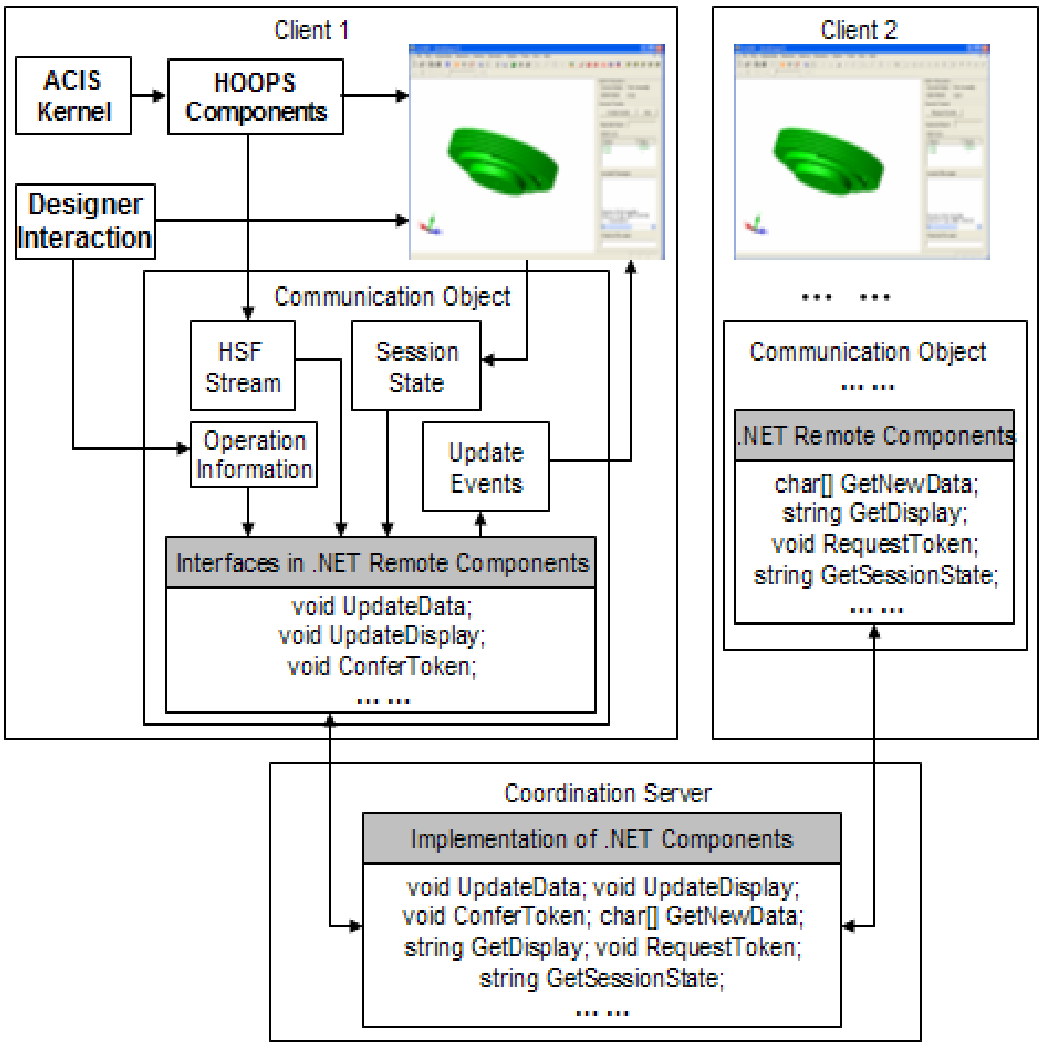


Figure 4.13: Remote .NET components in CoCADE System

Coordination Server provides .NET remote components to clients. These components include all the interfaces that are needed for collaborative activities. In figure 4.13, Client 1 is an active designer (token holder) who can edit the product model and update the latest information through the .NET remote components.

Other clients (Client 2 in 4.13) are observers who only receive the updated information or send requests for control token through the .NET remote components.

Two .Net components are implemented to realize coordination and synchronization. One is Session Manager component that is responsible for collaborative session management. The other is Message Handler that is responsible for synchronization during collaboration process. Table 4.1 lists the main functions of the two .NET components.

Table 4.1: Main functions of .NET components

Session Manager Interfaces	Functions
CreateSession	Create a new collaborative session
JoinSession	Join a existing collaborative session
LeaveSession	Leave a collaborative session
TerminateSession	Terminate a collaborative session

Message Handler Interfaces	Functions
UpdateData	Update HSF representation data
GetNewData	Get new representation data
UpdateDisplay	Update operation information
GetDisplay	Get new operation information
GetSessionState	Get new session state
RequestToken	Request operation token
ReleaseToken	Release operation token
Confertoken	Confer operation token to a client

4.2.5 Operation Delay

Collaborative applications can address the latency in term of response time [75]. CoCADE system utilizes the Internet as the medium for collaboration. It is important to consider the actual response time for different activities performed utilizing the CoCADE system.

In CoCADE system, response time is the time between a designer's operation and a remote collaborator's seeing the results of that operation. It is function of: (i) the available bandwidth of the Internet, (ii) the execution time of operation, and (iii) the operation message size. Of the three factors affecting the response time, the available bandwidth, which is resulted in queueing delay, has the maximum influence on the response time and cannot be predicted in advance due to rapid fluctuations of the available bandwidth. Therefore, in this section, the response time is categorized based on the type of data transferred through the Internet. The queueing delay is analyzed using a simplified queueing model.

- **Response Time**

A normal scenario is considered for calculating the response time (Figure 4.14): token holder generate an operation. Data holder executes the operation, then broadcasts visualization information to other clients through coordination server. Token holder and data holder reside at different client sites. Thus, there are two types of operations: Original Operation, which is executed by data holder based on ACIS geometric data, and HSF Operation, which is executed by other clients based on Hoops data for visualization only.

The average transaction time of Original Operation between client and server is t_1. The average transaction time of HSF Operation between client and server is t_2. The execution time of Original Operation is t_{e1}. The execution time of HSF Operation is t_{e2}.

The response time for creation of the primitives (block, cylinder, sphere etc) is the time required for completion of the following operations: sending Original Operation to data holder, executing Original Operation, sending HSF operation to other clients (token holder in this scenario), executing HSF operation. It can be calculated by: $2t_1+2t_2+t_{e1}+t_{e2}$.

The response time for loading models requires the following operations to be completed: data holder loads ACIS based data, sending HSF representation data to token holder, Visualizing representation data. It can be calculated by: $2t_2+t_{e1}+t_{e2}$.

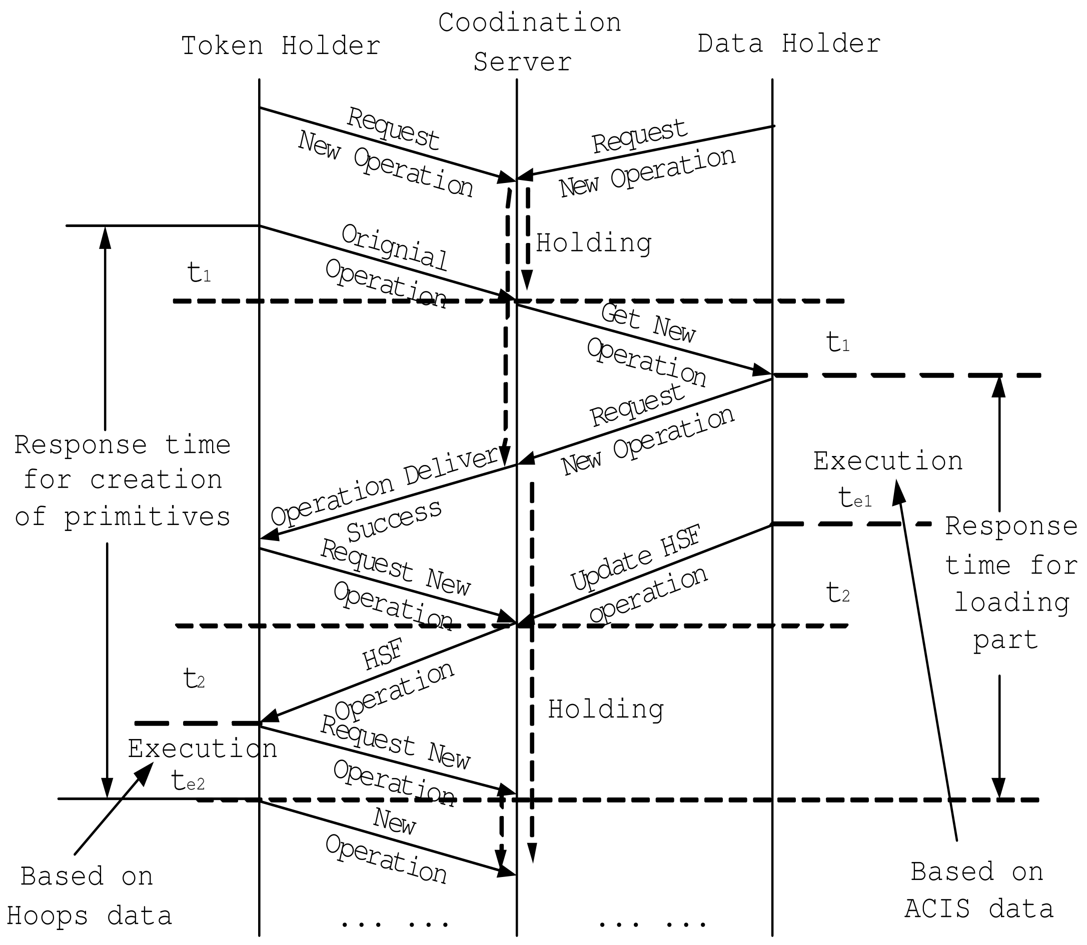


Figure 4.14: Response time under normal conditions

During these trials, the server was running on a computer having Intel Pentium IV 2.66GHz CPU; The client was executed on another PC, which had an Intel Pentium IV 1.5GHz CPU. The test part for measuring response time are shown in Figure 4.15. The execution time and operation message size can be read from client program as shown in Table 4.2.

Table 4.2: Execution time and operation message length

Operation	Part	Original Operation length (byte)	*Execution T_{e1} (ms)	HSF Operation length (byte)	*Execution T_{e2}
Create	Sphere (a)	65	50	133	10
Create	Block (b)	81	70	145	10
Create	Cylinder (c)	77	60	142	10
Load	Plate (d)	NA	50	1849	10
Load	Clamper (e)	NA	250	9014	30
Load	Gear (f)	NA	530	16718	60

* The timer accuracy is 10 ms for client Windows XP operating system

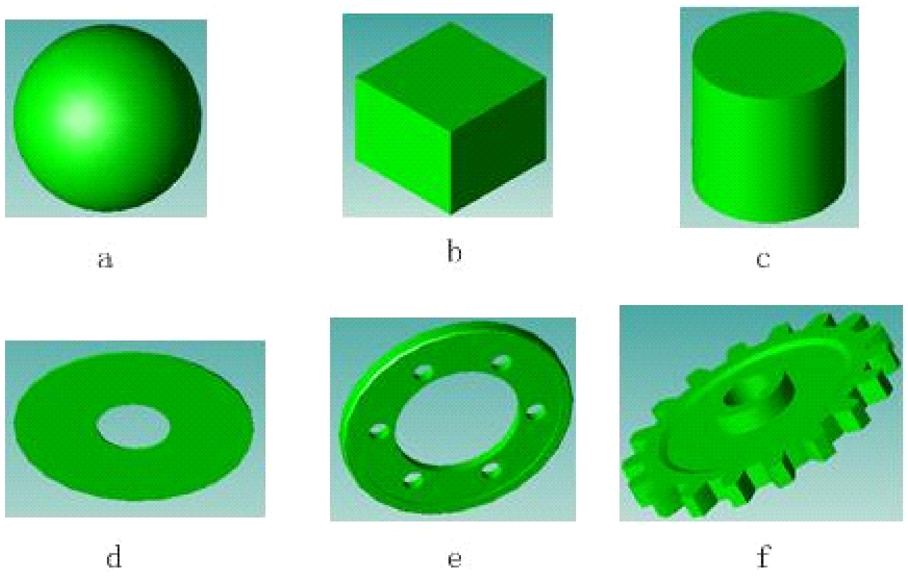


Figure 4.15: Test parts for measuring response time

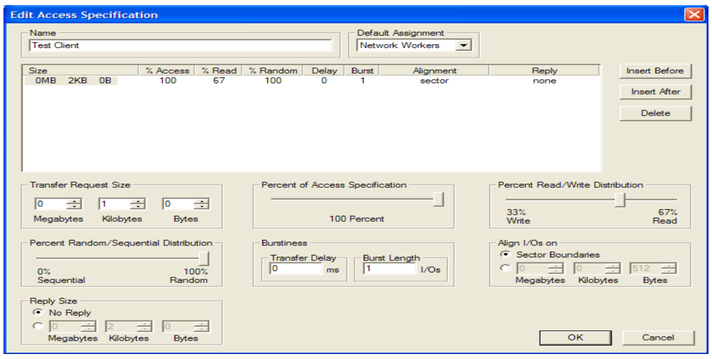
Response Time Using TCP/IP Connection

To obtain the message transaction time between client and server, Iometer [76], the popular I/O subsystem measurement and characterization tool, is adopted. The remote access specification is defined (A in Figure 4.16) and a distributed environment with ten clients (B in Figure 4.16) is simulated. The size of operation information package for transmission is set to the same as that are listed in Table 4.2. The package size of parts representation data is set to the same as HSF package size, that is 8KB each package. Then the average transaction time can be read from Iometer Results Display Panel (C in Figure 4.16). The response time of operation using TCP/IP connection is calculated in Table 4.3.

Table 4.3: Response time using TCP/IP conncection

Operation	Part	Original Operation length/Delay T_1 (byte/ms)	*Execution T_{e1} (ms)	HSF Operation length/Delay T_2 (byte/ms)	*Execution T_{e2} (ms)	Response time (ms)
Create	Sphere (a)	65/10	50	133/11	10	102
Create	Block (b)	81/10	70	145/11	10	122
Create	Cylinder (c)	77/10	60	142/11	10	112
Load	Plate (d)	NA	50	1849/15	10	88
Load	Clamper (e)	NA	250	9014/41	30	352
Load	Gear (f)	NA	530	16718/64	60	706

* The timer accuracy is 10 ms for client Windows XP operating system



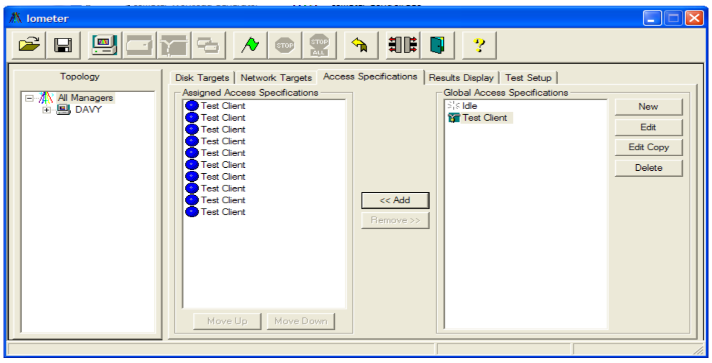
Edit Access Specification

Name: Test Client
Default Assignment: Network Workers

Size	% Access	% Read	% Random	Delay	Burst	Alignment	Reply
0MB 2KB 0B	100	67	100	0	1	sector	none

Transfer Request Size: 0 Megabytes, 1 Kilobytes, 0 Bytes
Percent of Access Specification: 100 Percent
Percent Read/Write Distribution: 33% Write, 67% Read
Percent Random/Sequential Distribution: 0% Sequential, 100% Random
Burstiness: Transfer Delay 0 ms, Burst Length 1 I/Os
Align I/Os on: [✓ Sector Boundaries]; 0 Megabytes, 0 Kilobytes, 512 Bytes
Reply Size: [✓ No Reply]; 0 Megabytes, 2 Kilobytes, 0 Bytes

A. Edit remote access specification



Iometer

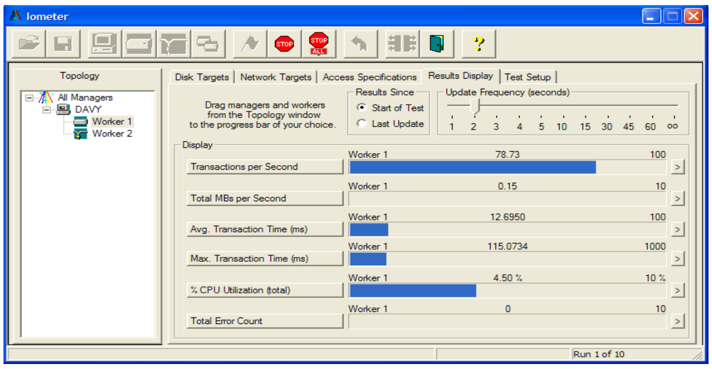
Topology: All Managers, DAVY

Disk Targets | Network Targets | Access Specifications | Results Display | Test Setup

Assigned Access Specifications: Test Client, Test Client, Test Client, Test Client, Test Client, Test Client, Test Client, Test Client, Test Client, Test Client

Global Access Specifications: Idle, Test Client

B. Setup remote access clients



Iometer

Topology: All Managers, DAVY, Worker 1, Worker 2

Disk Targets | Network Targets | Access Specifications | Results Display | Test Setup

Drag managers and workers from the Topology window to the progress bar of your choice.

Results Since: [✓ Start of Test] Last Update
Update Frequency (seconds): 1 2 3 4 5 10 15 30 45 60 oo

Display

Transactions per Second	Worker 1	78.73	100
Total MBs per Second	Worker 1	0.15	10
Avg. Transaction Time (ms)	Worker 1	12.6950	100
Max. Transaction Time (ms)	Worker 1	115.0734	1000
% CPU Utilization (total)	Worker 1	4.50 %	10 %
Total Error Count	Worker 1	0	10

Run 1 of 10

C. Read transaction time from results display panel

Figure 4.16: Transaction time obtained using Iometer

Response Time Using HTTP Connection

For HTTP connection, data for transmission is described by SOAP message which is generated by .NET Remoting class for transmission over internet. Thus, additional text message is added because of using SOAP. A typical SOAP message generated by .NET framework is shown in Figure 4.17:

To obtain a formal solution for calculating response time, the average SOAP head is set as 512 bytes. Then the average transaction time is measured using Iometer and the average response time is calculated (Table 4.4).

```
<SOAP-ENV:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0"
  SOAP-ENV:encodingStyle=
  "http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:Method
   xmlns: m= "http://server/Method.soap>
   <messagebody>message</messagebody>
  <m:Method>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 4.17: Soap message for invoking remote object

Table 4.4: Response time using HTTP conncection

Operation	Part	Original Operation length/Delay T_1 (byte/ms)	*Execution T_{e1} (ms)	HSF Operation length/Delay T_2 (byte/ms)	*Execution T_{e2} (ms)	Response time (ms)
Creation	Sphere (a)	577/15	50	645/16	10	122
Creation	Block (b)	593/15	70	657/16	10	142
Creation	Cylinder (c)	589/15	60	654/16	10	132
Load	Plate (d)	NA	50	2361/34	10	128
Load	Clamper (e)	NA	250	9526/71	30	422
Load	Gear (f)	NA	530	17230/121	60	832

* The timer accuracy is 10 ms for client Windows XP operating system

- **Queuing Delay**

For the interconnected network, where client and server reside in different LANs, queuing is inevitable for routing or flow control (Figure 4.18). The queuing delay is one of the most important factors that affects the response time.

To obtain a better understanding of queuing delay, M/M/1 queueing model [77] is adopted to represent the queuing system. The M/M/1 queuing system consists of a single transmission line. Data packages arrive according to a Poisson process with the rate λ, and the transmission rate μ. The average transmission time is $\tau = 1/\mu$. The utilization factor of transmission line can be denoted as

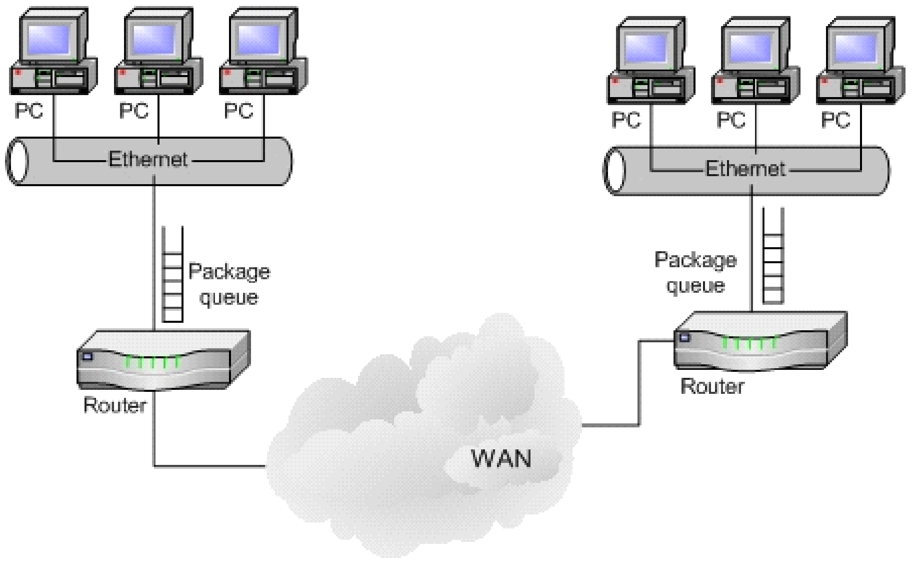


Figure 4.18: Queuing delay in interconnected network

$$\rho = \lambda/\mu \tag{4.1}$$

The average number of packages in the system in steady-state is:

$$N = \lambda/(\mu - \lambda) \tag{4.2}$$

The average delay per package (waiting time in queue plus transmission time) is given by By Little's Theorem [78],

$$T = N/\lambda \tag{4.3}$$

Using equation (4.1) and (4.2), this becomes,

$$T = 1/(\mu - \lambda) = \tau/(1 - \rho) \tag{4.4}$$

The average waiting time in queue, W, is the average delay T less the average transmission time $1/\mu$, therefore,

$$W = 1/(\mu - \lambda) - 1/\mu = \rho\tau/(1 - \rho) \tag{4.5}$$

From equation (4.4) and (4.5), it can be concluded that the utilization of the transmission line has heavy impact on the transmission delay. When $\rho \to 0$, $T \approx \tau$, that is when transmission rate is far higher than package arriving rate, the average delay per package is near the transmission time. When ρ increases, $T > \tau$, that is when package arriving rate increases, the average delay per package increases accordingly. When $\rho \to 1$, $T \to \infty$, that is when packaging arriving rate is near or exceed the transmission rate, network congestion will occur. The package may be blocked or discarded.

Based on the discussion in this section, it is clear that transferring initial representation data is a slower process compared with transferring other operation information. This transfer typically occurs when a new session is established. For example, when a new session starts, product part is loaded into workplace for editing. In order to minimize the transfer of large CAD based representations of the product, HSF operation is used to synchronize client views. The response time using HTTP connections is higher than that using TCP/IP connections. It is partially because of the queueing delay. Although it is difficult to predict available network bandwidth, the M/M/1 queuing delay model provide a basis for adequate delay approximations.

4.3 Summary

In this chapter, a workflow-driven mechanism has been proposed to manage collaborative sessions in CoCADE framework. It is found that the workflow system can dynamically define and organize collaborative sessions during product design process to facilitate product design activities. The critical issues, such as coordination and synchronization, which have important impact on the synchronous collaborative design efficiency, have been studied in detail. CCM mechanism and MTRR scheme have been proposed to provide reliable support for collaborative design and analysis in synchronous collaborative session.

Chapter 5

Case Study - Spindle Motor Design and Analysis

In this chapter, an electromagnetic design case relating to spindle motors for hardisk drives is presented to demonstrate how a collaborative design and analysis work is supported by CoCADE framework with the collaborative session management scheme.

5.1 Introduction to Product Design

Ulrich and Eppinger [79] have defined product development as “the set of activities beginning with the perception of a market opportunity and ending in the production, sale and delivery of product”. These activities form a product lifecycle from concept specification, through design and manufacture to disposal (Table 5.1).

Table 5.1: Stages in product lifecycle

	Product Development Cycle	Product Delivery Cycle
1	Customer requirements	Process planning
2	Product specification	Manufacture
3	Conceptual design	Inspection
4	Detailed design	Sale
5	Simulation and Optimization	Support and Services
6	Prototyping and Testing	Disposal and Recycle

Product Lifecycle Management (PLM) covers all stages of product lifecycle and has been widely recognized as a strategic business approach to support a product-centric business solution that unifies product lifecycle by enabling online sharing of product knowledge and business applications [80, 81, 82, 83]. In the product development cycle, the need for inner-enterprise and inter-enterprise collaborations becomes more and more intense. As shown in Figure 5.1, it is very important to bring engineers from relevant fields together to solve design problems in the early design stage. Better collaboration of the designers working on a common design task will reduce the product development lead-time to a large extent.

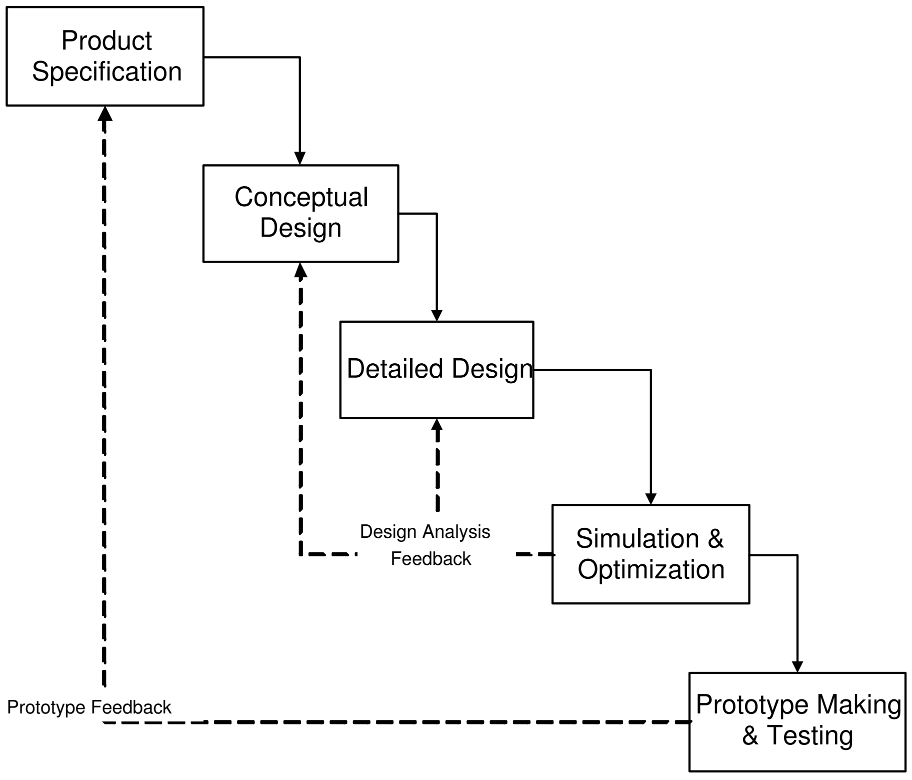


Figure 5.1: Product development cycle

5.2 Introduction to Spindle Motor

Hard disk drive is one of the most important components within a PC. Using the magnetic recording technologies, the hard disk drive is capable of storing large amounts of digital information. The spindle motor (Figure 5.2) is a key component in the hard disk drive, as it is responsible for turning the hard disk platters, and allowing the hard drive to operate. For accurate recording of information, it is extremely important for the platter to revolve at a constant speed with as little vibration as possible [84]. The spindle motor must provide stable and reliable turning power for thousands of hours of often continuous use.

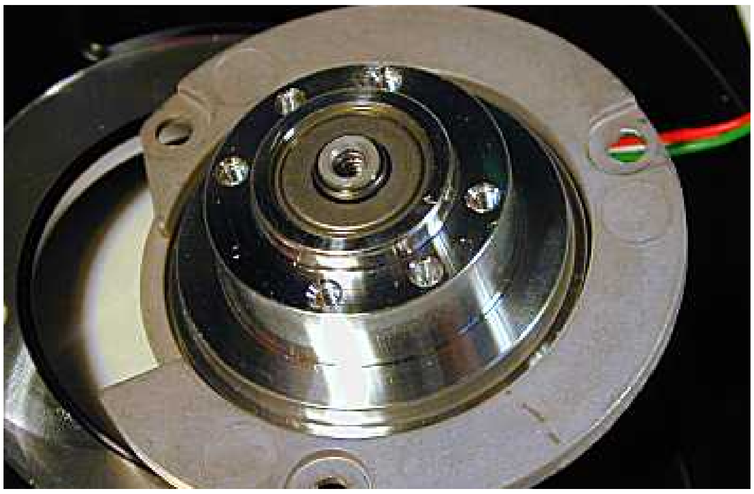


Figure 5.2: A hard disk spindle motor

The precise circular motion is determined by the torque, which is produced by the magnetic field in the hard disk spindle motor. Therefore, an accurate analysis of the magnetic field is necessary in design stage. In order to predict instantaneous magnetic field distribution in the spindle motor, numerical technique based on the Finite Element Analysis (FEA) has been adopted [85]. It is a process which takes a CAD model as input and adds materials, applies loads and constraints to the model, and simulate the performance. Through dividing the whole problem domain into finite small regions in which the field problem can be approximately represented by linear equations, FEA is perhaps the most useful numerical tool to solve non-linear field problems with complex geometry.

5.3 Spindle Motor Design using CoCADE Framework

Figure 5.3 shows a spindle motor design scenario based on the CoCADE framework with workflow-driven collaborative session management mechanism.

Assume that Engineers A, B (Company I) and C (Company II) involves in the collaborative design of a spindle motor using the CoCADE system. CoCADE servers are deployed in Company II. They will follow the major steps discussed in the following sections.

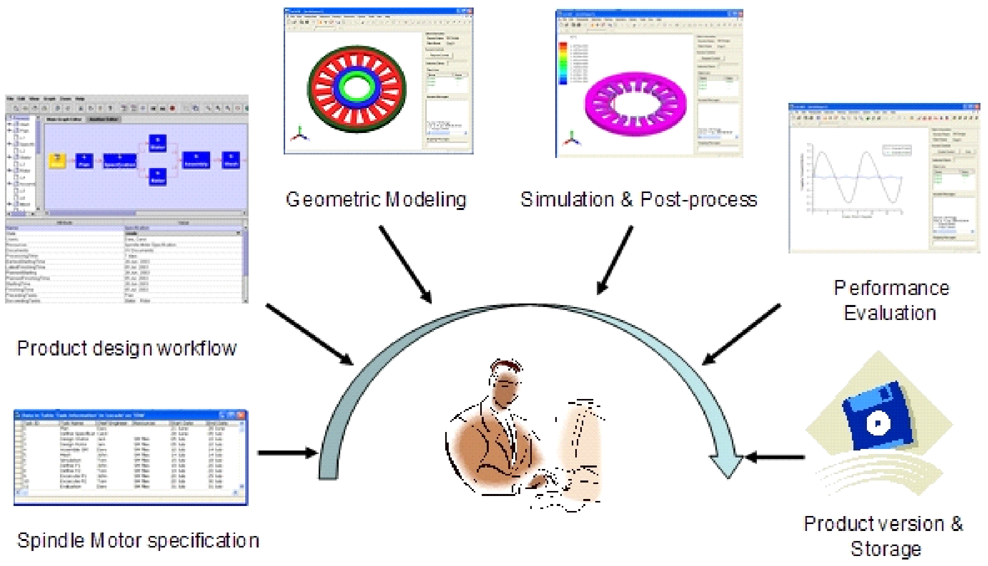
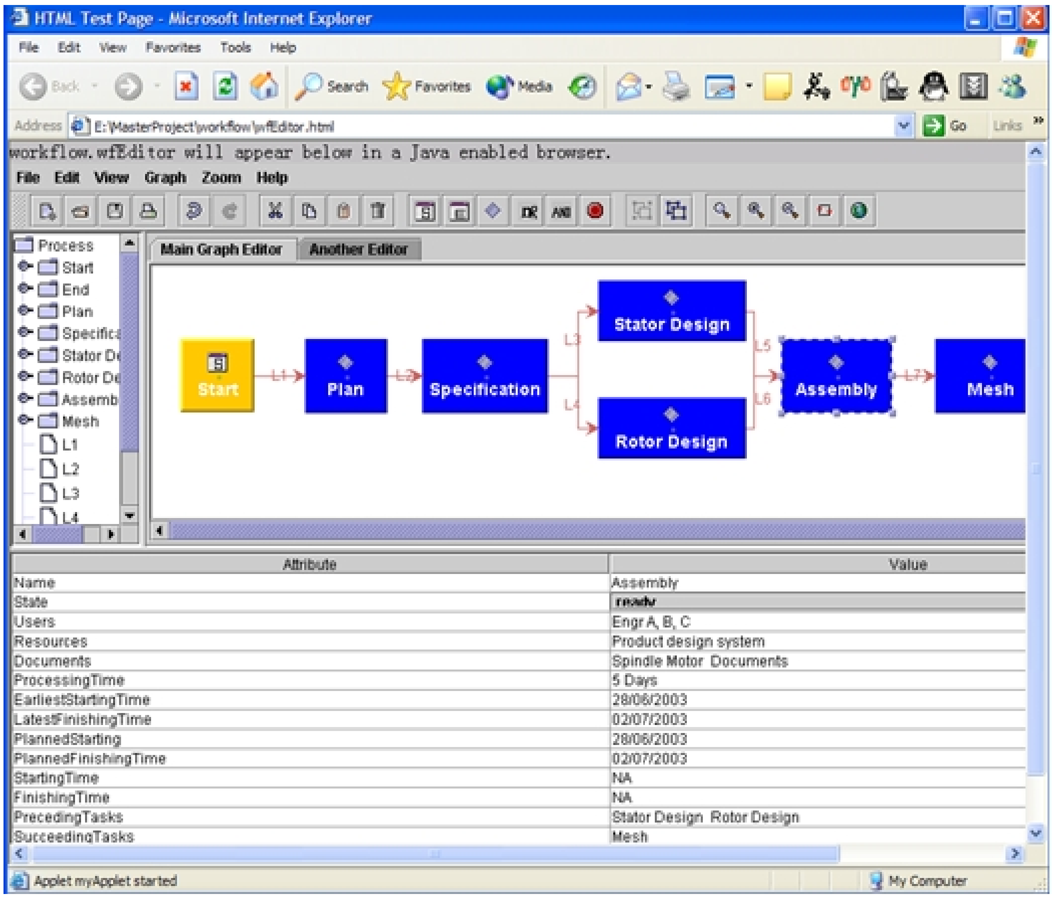

Geometric Modeling
Simulation & Post-process
Performance Evaluation
Product design workflow
Spindle Motor specification
Product version & Storage

Figure 5.3: A collaborative spindle motor design scenario

5.3.1 Product Design Process Definition

Planning is the first step for the product design process. The workflow process is implemented with consideration of the following:

- The requirements of product design and product specification.
- Appropriate knowledge that can be used to enhance the workflow process.


HTML Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address E:\MasterProject\workflow\wfEditor.html
workflow.wfEditor will appear below in a Java enabled browser.
File Edit View Graph Zoom Help
Process
Start
End
Plan
Specifica
Stator De
Rotor De
Assemb
Mesh
L1
L2
L3
L4
Main Graph Editor
Another Editor
Start
L1
Plan
L2
Specification
L3
L4
Stator Design
Rotor Design
L5
L6
Assembly
L7
Mesh
Applet myApplet started
My Computer

Attribute	Value
Name	Assembly
State	ready
Users	Engr A, B, C
Resources	Product design system
Documents	Spindle Motor Documents
ProcessingTime	5 Days
EarliestStartingTime	28/06/2003
LatestFinishingTime	02/07/2003
PlannedStarting	28/06/2003
PlannedFinishingTime	02/07/2003
StartingTime	NA
FinishingTime	NA
PrecedingTasks	Stator Design Rotor Design
SucceedingTasks	Mesh

Figure 5.4: Define product design workflow model

- The most efficient and useful methods of integrating and transferring knowledge.
- The needs of distributed designers and the required modes of communication.

Based on the spindle motor specifications, a collaborative design workflow model is created by engineers A, B and C cooperatively through a product design workflow editor. As shown in Figure 5.4, the task attributes consisting of states, users, resources, documents, time requirements, etc., are described. Accordingly, collaborative sessions are defined with consideration of such task attributes. The possible dependencies are evaluated in the workflow model and corresponding solutions are described as new task. The workflow planner is to make effort to predict the potential interdependent tasks in product modeling stage and computing stage. This can avoid conflicts and errors occurred in product development process in earlier stage, and save time and resources. After completion of process definition, a

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href=""?>
<Process id="0" name="Spindle Motor Design Process" description="Case Study" state="0">
        <Users>A</Users>
        <Resources>CoCADE System, ANSYS, ANSOFT, SQL Server</Resources>
        <Documents>Spindle Motor Documents</Documents>
<Task id="1" taskType="1" rect="18,18,50,50" name="Plan" description="" state="0">
        <PrecedingTasks></PrecedingTasks>
        <SucceedingTasks>Spindle Motor Specification</SucceedingTasks>
        <Users>A</Users>
        <Resources>Product Design Workflow System</Resources>
        <Documents>Spindle Motor Documents</Documents>
</Task>
<Task id="2" taskType="1" rect="12,12,50,50" name="Specification" description="" state="0">
        <PrecedingTasks>Project Planning</PrecedingTasks>
        <SucceedingTasks>Stator Design,Rotor Design</SucceedingTasks>
        <Users>A,B,C</Users>
        <Resources>SQL Server</Resources>
        <Documents>Spindle Motor Documents</Documents>
</Task>
... ...
</Process>
```

Figure 5.5: Design process defined by XML

XML file is created by workflow engine to record all necessary information relating to the design process. Figure 5.5 shows a sample of XML file.

Based on the above process definition, the spindle motor design and analysis flow in this study is illustrated in Figure 5.6. collaborative sessions are established to facilitate the collaborative design activities. In asynchronous collaborative session, engineers can design product parts asynchronously and collaboratively. For example, the stator and rotor design (Figure 5.6) are carried out in asynchronous sessions. Synchronous collaborative session requires that all participants join the same workplace and carry out the same design task in real time. It requires coordination and synchronization in the collaboration process. For example, the Performance Evaluation task (Figure 5.6) are carried out in a synchronous collaborative session.

5.3.2 Product Modeling

Following the pre-defined workflow model, collaborative design can be carried out by engineers from different disciplines. The design tasks can be performed by multiple engineers in parallel. As illustrated in Figure 5.7, the stator and rotor are

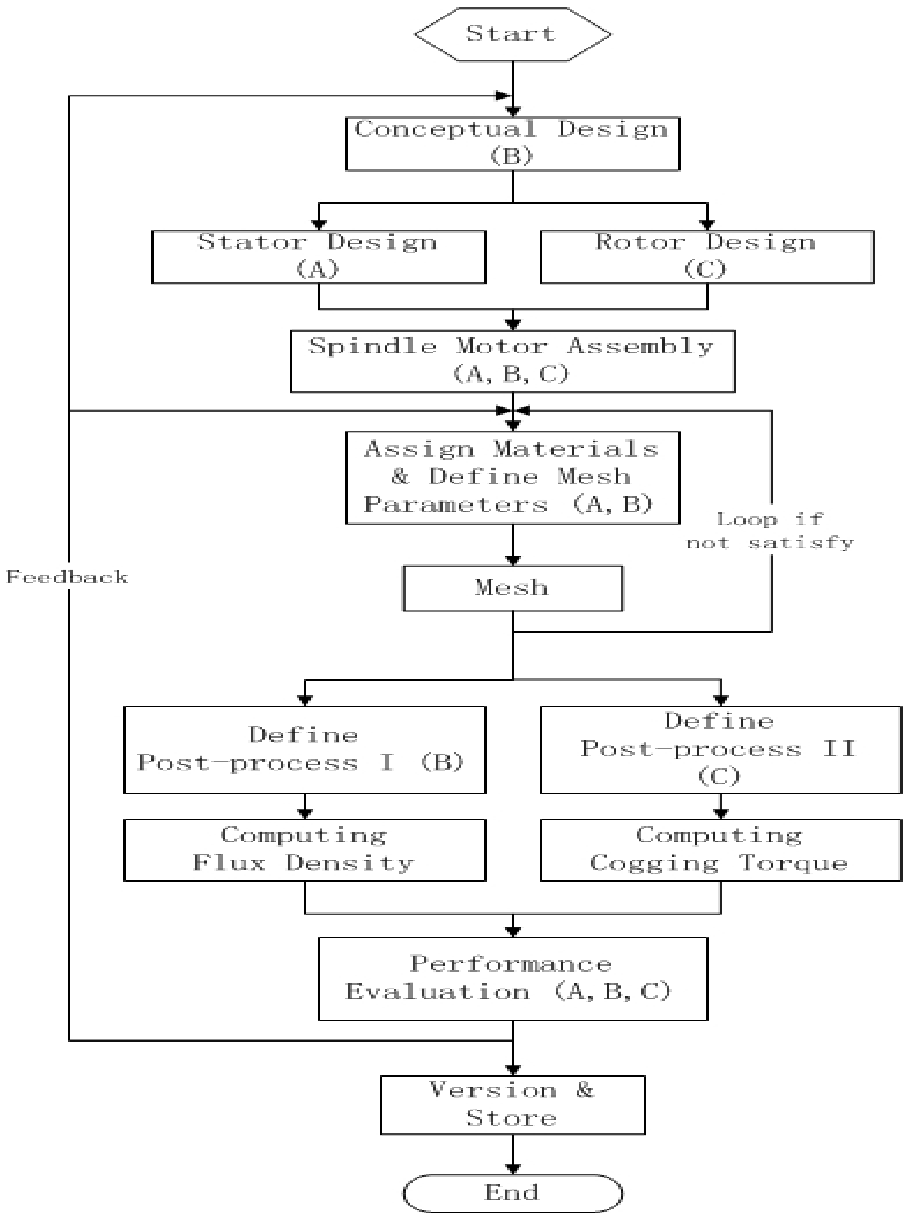

Start
Conceptual Design (B)
Stator Design (A)
Rotor Design (C)
Spindle Motor Assembly (A, B, C)
Assign Materials & Define Mesh Parameters (A, B)
Loop if not satisfy
Feedback
Mesh
Define Post-process I (B)
Define Post-process II (C)
Computing Flux Density
Computing Cogging Torque
Performance Evaluation (A, B, C)
Version & Store
End

Figure 5.6: Spindle motor design and analysis flow

designed by Engineer A (Company I) and C (Company II) asynchronously. The asynchronous collaborative design flow is:

- Engineer A checks out stator model from public database reside at server side. The accessibility of the stator model from other engineers will then be denied. Coordination server sends updating message to workflow engine and the new task status is set to “Start”.

- Engineer A creates a asynchronous collaborative session in CoCADE system and perform geometric design. Workflow engine will record the duration of

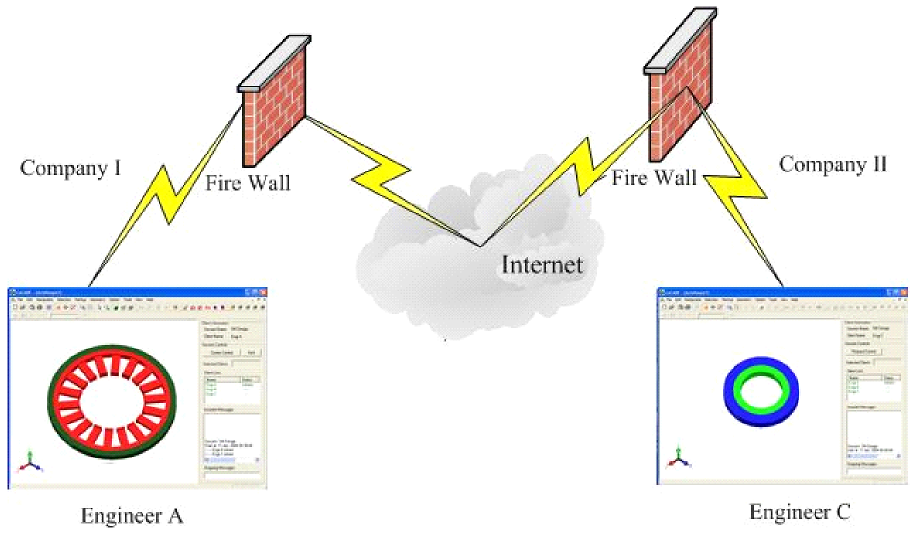
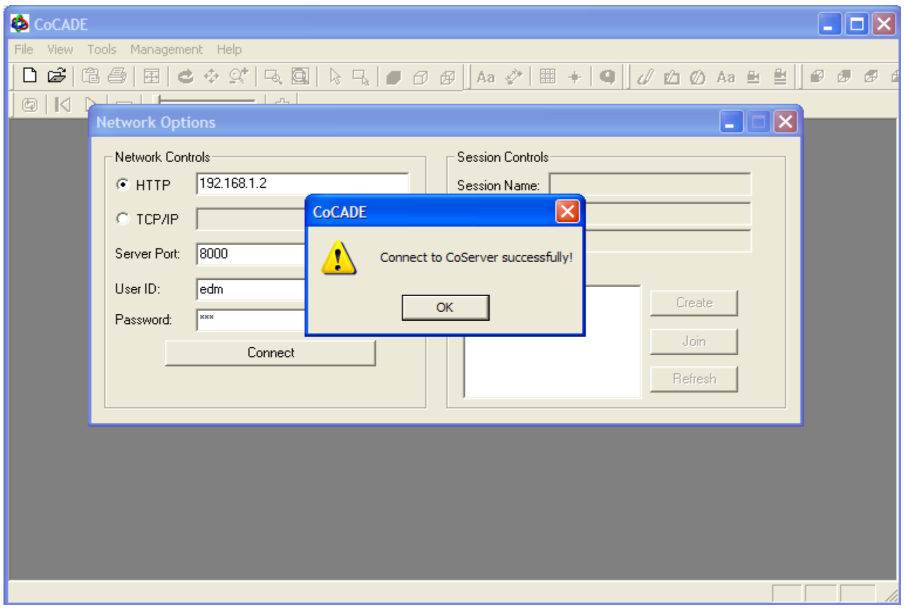


Figure 5.7: Asynchronous collaborative session

this design task. Simultaneously, Engineer C can perform the rotor design in the session.

- When finishing stator design, Engineer A checks in the stator model as a new version. The accessibility of stator model is granted to all authorized engineers.
- Engineer C checks out the latest version of stator model and perform some modifications.
- Upon completion of the stator and rotor design, the workflow state will move to the next task. The new task status of stator and rotor design is set to "End".

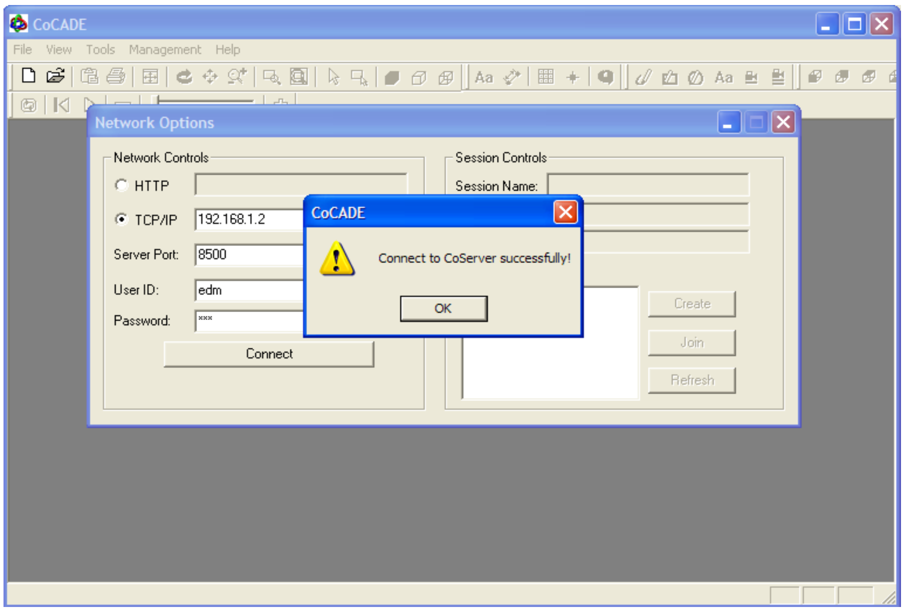
The involved participants also need to create synchronous collaborative sessions to carry out the same design task in the same workplace, such as the Spindle Motor Assembly task in Figure 5.6. Engineers A, B and C establish the synchronous collaborative session for assembling spindle motor and discuss on the model with the help of coordination server that are implemented with the coordination protocol and synchronization scheme. The flow of synchronous collaborative design is:



HTTP: 192.168.1.2
TCP/IP:
Server Port: 8000
User ID: edm
Password: ***

Connect to CoServer successfully!

i: HTTP Connection



HTTP:
TCP/IP: 192.168.1.2
Server Port: 8500
User ID: edm
Password: ***

Connect to CoServer successfully!

ii: TCP/IP Connection

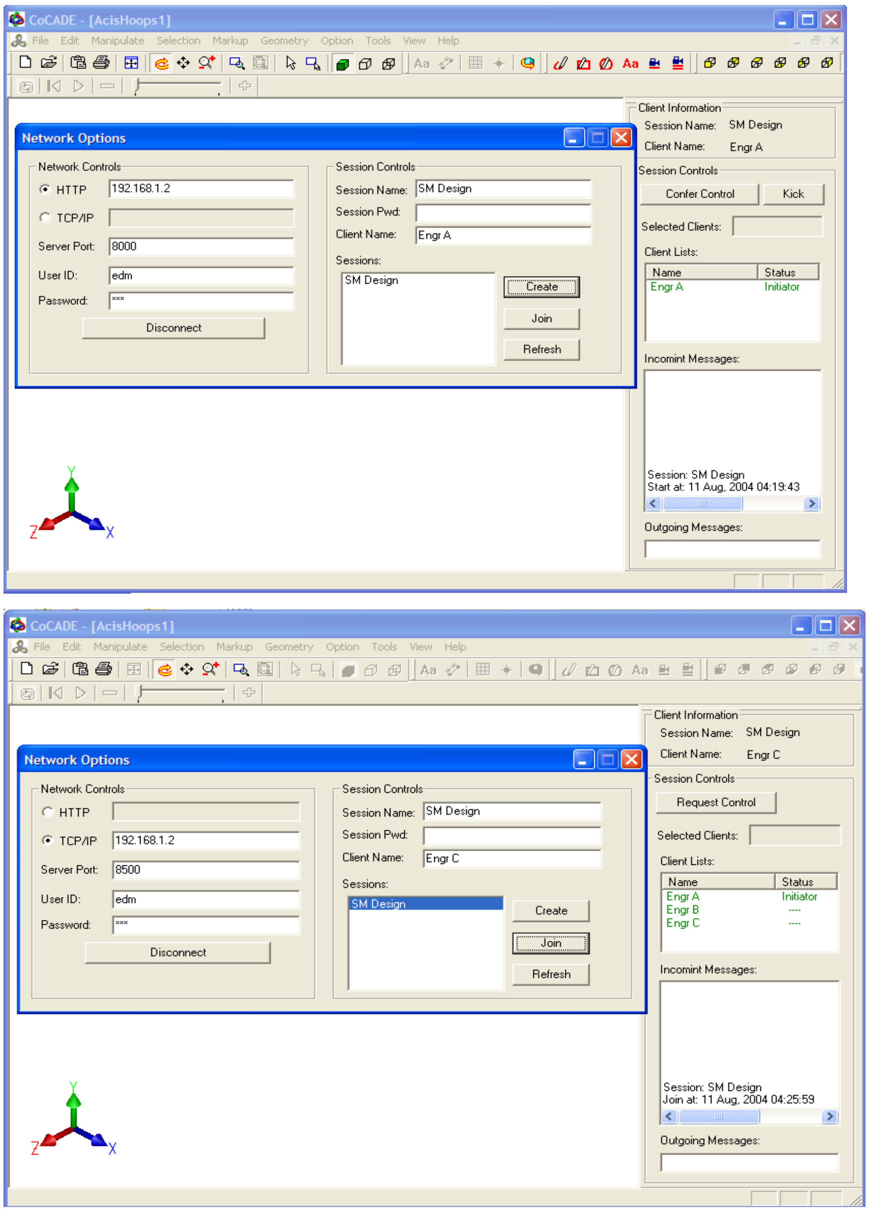
Figure 5.8: Connect to server using HTTP or TCP/IP

- **Connect to server:** As Engineer A and B are out of Company II, they use HTTP to connect CoCADE server (i in Figure 5.8). While Engineer C uses TCP/IP to connect CoCADE server for the best performance (ii in Figure 5.8).

- **Establish session:** Engineer A creates the synchronous collaborative session named "SM Design" (i in Figure 5.9), checkouts the latest version of the stator and rotor. Then Engineer A waits for other engineers. Engineer B and C to join the session (ii in Figure 5.9). Engineer B and C receive the initial representation data that Engineer A has created or loaded from Coordination Server, in this case the part of a spindle motor.

i: Create Session

ii: Join Session


CoCADE - [AcisHoops1]

Network Options

Network Controls: HTTP (selected) 192.168.1.2; TCP/IP; Server Port: 8000; User ID: edm; Password: ***

Session Controls: Session Name: SM Design; Session Pwd:; Client Name: Engr A; Sessions: SM Design

Client Information: Session Name: SM Design; Client Name: Engr A

Client Lists:

Name	Status
Engr A	Initiator

Incomint Messages:
Session: SM Design
Start at: 11 Aug, 2004 04:19:43

CoCADE - [AcisHoops1]

Network Options

Network Controls: HTTP; TCP/IP (selected) 192.168.1.2; Server Port: 8500; User ID: edm; Password: ***

Session Controls: Session Name: SM Design; Session Pwd:; Client Name: Engr C; Sessions: SM Design

Client Information: Session Name: SM Design; Client Name: Engr C

Client Lists:

Name	Status
Engr A	Initiator
Engr B	----
Engr C	----

Incomint Messages:
Session: SM Design
Join at: 11 Aug, 2004 04:25:59

Figure 5.9: Create session and join session

- **Collaborative design (Figure 5.10):** Collaborative communication [86] takes place after the establishment of collaborative session. Engineer A, B, and C discuss on the product model and share their ideas with the help of CoCADE collaboration tools, such as instance messaging tool, mark-up tool etc. They can also modify the product model cooperatively under CCM (Centralized Coordination Mechanism). One can manipulate the product model when he obtains the control token from previous token holder. The initiator, Engineer A, is the session chairman who can keep the collaboration process in a controlled and cooperative manner. Operations, initial representation data and session status are synchronized using MTRR (Multi-thread Request Response scheme).

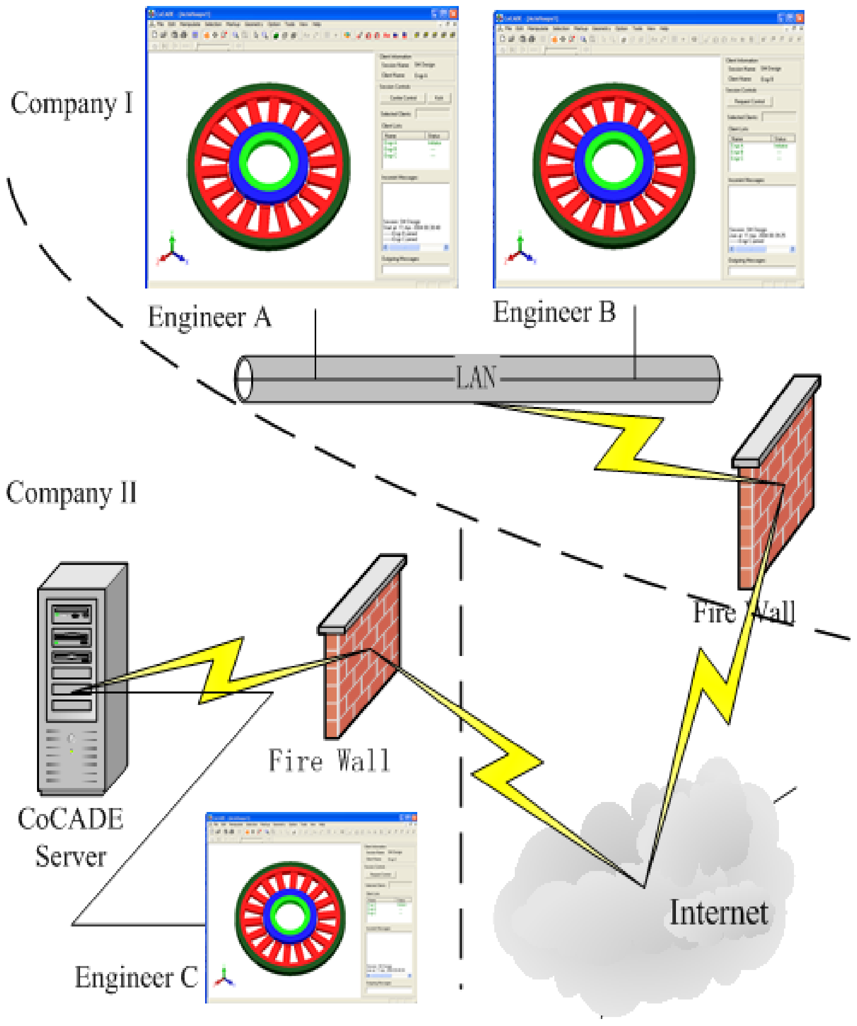
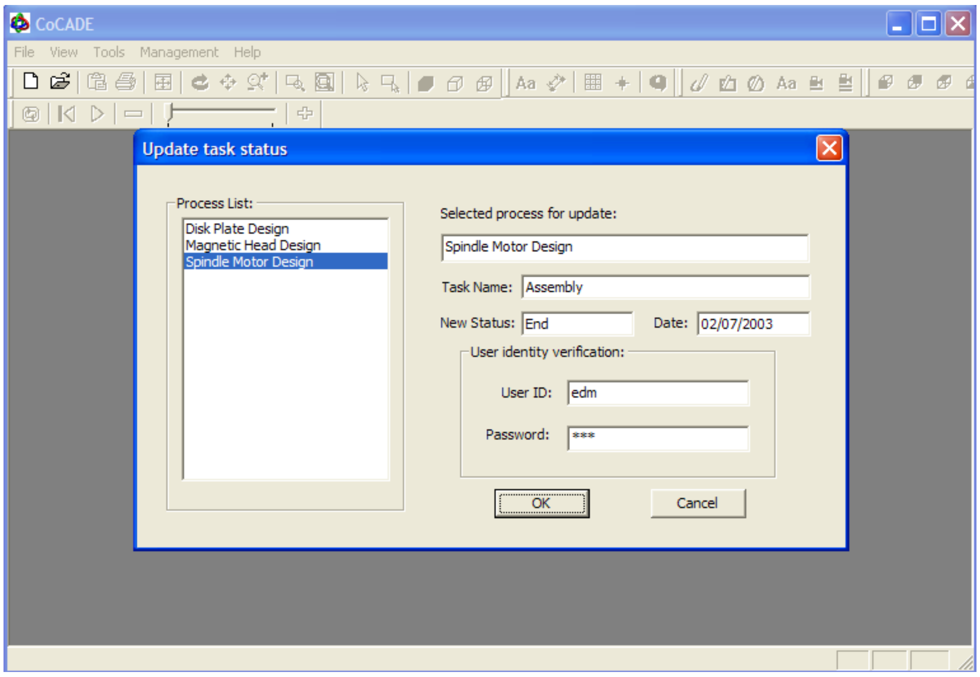


Figure 5.10: Synchronous collaborative session

- **Save product model:** After Engineer A, B and C reach an agreement of spindle motor geometry, Engineer A checks in the product model to public database.

- **Update task information :** After completion of a design task, the corresponding information is updated through coordination server to the workflow model (Figure 5.11). Then the product design will proceed to the next task.



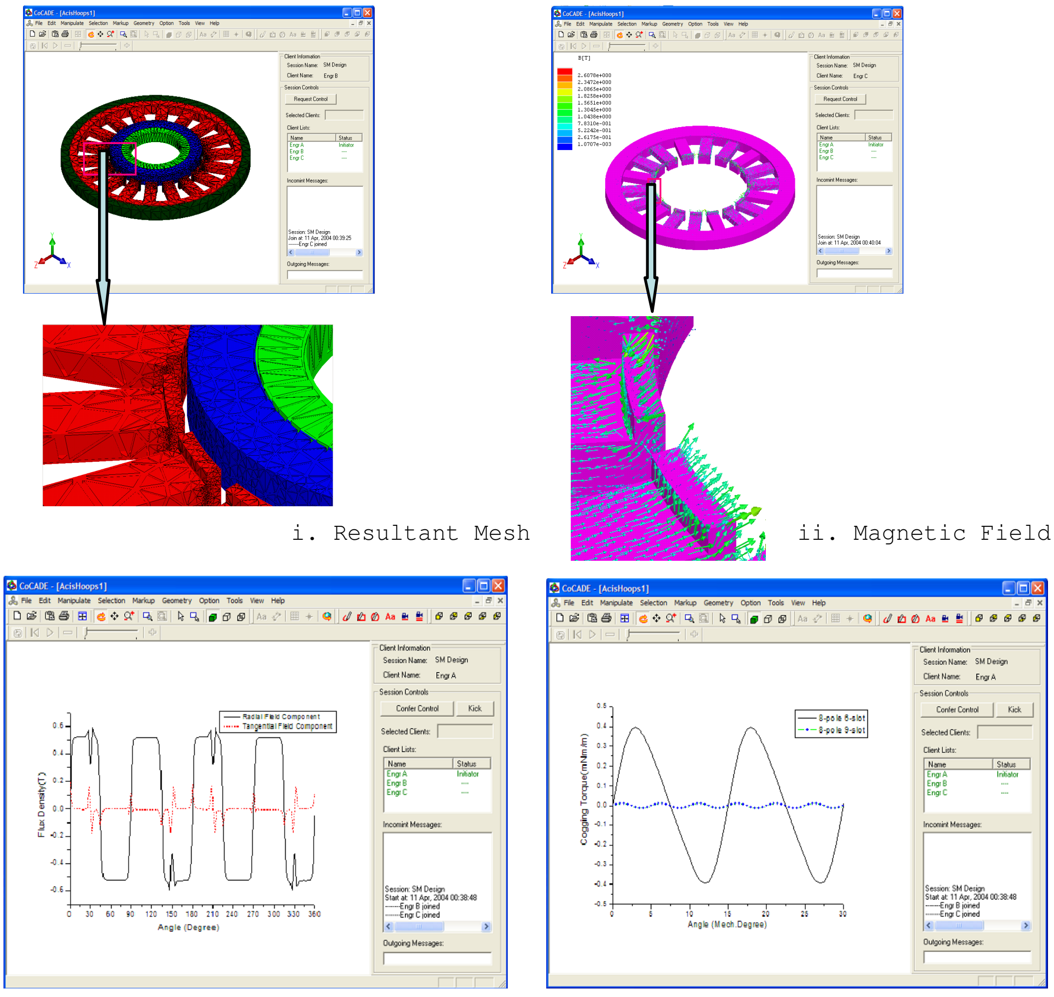
Process List:
Disk Plate Design
Magnetic Head Design
Spindle Motor Design

Selected process for update: Spindle Motor Design
Task Name: Assembly
New Status: End
Date: 02/07/2003
User identity verification:
User ID: edm
Password: ***

Figure 5.11: Update task information

5.3.3 Product Performance Evaluation

Having finished product design process, three engineers can discuss and decide the data preparation for simulation purpose including mesh generation and assignment of materials properties, sources/load, and boundary condition. Such information is sent to CAE server to solve the associated physical problems. Engineers can discuss about the resultant mesh (i in Figure 5.12) and re-define mesh parameters until desirable results (ii in Figure 5.12) are obtained. After the completion of simulation process, engineers collaboratively define post-process parameters and send these parameters to CAE server. The instruction handler that is implemented in CAE server generates the corresponding macroinstruction stream to invoke computing process of CAE solvers. Figure 5.13 shows a sample macroinstruction stream for computing cogging torque.



i. Resultant Mesh　　ii. Magnetic Field

iii. Flux Density　　iv. Cogging Torque

Figure 5.12: Product performance evaluation

In this study, the definition of computing tasks (flux density and cogging torque), which are performed by Engineer B and C respectively, are carried out in asynchronous sessions (Figure 5.6). However, the computing of cogging torque requires information on computed flux density obtained from post-process I. The workflow engine will detect dependencies and their interfaces to avoid conflicts among these computing tasks.

In order to find the slot effect on the distribution of the flux density in the spindle motor, three engineers decide to evaluate the flux density generate by post-process I before computing cogging toque. A new task, flux density evaluation,

```
!====CALCULATE COGGING TORQUE====
!       T=L*(R^2)*INTERGRATE(BN*BT)d(theta)/MIU0
!       L IS THE AXIAL LENGTH OF MOTOR      R IS THE CALCULATION RADIUS IN AIRGAP
!       BN AND BT ARE NORMAL AND TANGENTIAL COMPONETN OF FLUX DENSITY
!       d(theta)=2*PI/720
!================================

COG=0
*DIM,SDBRDATA,ARRAY,720                    !SAVE FLUX DENSITY DATA TO FILE
*DIM,SDBTDATA,ARRAY,720
*DIM,SDVPDATA,ARRAY,720

*DO,J,1,2*ROTA
       COG=COG+5*RC*RC*BNTDATA(J,5,1)*BNTDATA(J,6,1)/720
       NUMBR(J)=BNTDATA((719-2*ROTA+J),5,1)
       NUMBT(J)=BNTDATA((719-2*ROTA+J),6,1)
*ENDDO
*DO,J,(2*ROTA+1),720
       COG=COG+5*RC*RC*BNTDATA(J,5,1)*BNTDATA(J,6,1)/720
       NUMBR(J)=BNTDATA((J-2*ROTA),5,1)
       NUMBT(J)=BNTDATA((J-2*ROTA),6,1)
*ENDDO


/OUTPUT,COG0,DAT
*STATUS,COG

/OUTPUT,NUMBR,DAT
*STATUS,NUMBR                       !SAVE FLUX DENSITY DATA TO FILE
/OUTPUT,NUMBT,DAT
*STATUS,NUMBT,DAT

!/OUTPUT,SDVPDATA,DAT
!*STATUS,SDVPDATA
!*MWRITE,BRDATA,BR,DAT
```

Figure 5.13: Macroinstruction stream for computing cogging torque

is inserted into the workflow model dynamically. After verification of workflow engine, the workflow model is updated as shown in Figure 5.14.

The result of flux density is downloaded by Engineer A. Engineer B and C can obtain the updated representation data from coordination server. The radical component of the flux density and the tangential component are analyzed by three engineering synchronously (iii in Figure 5.12).

After completion of the new task, the cogging torque is calculated and the computational result is viewed by three engineers. They compare the cogging torque with previous version of spindle motor and cooperatively analyze the spindle motor performance (iv in Figure 5.12).

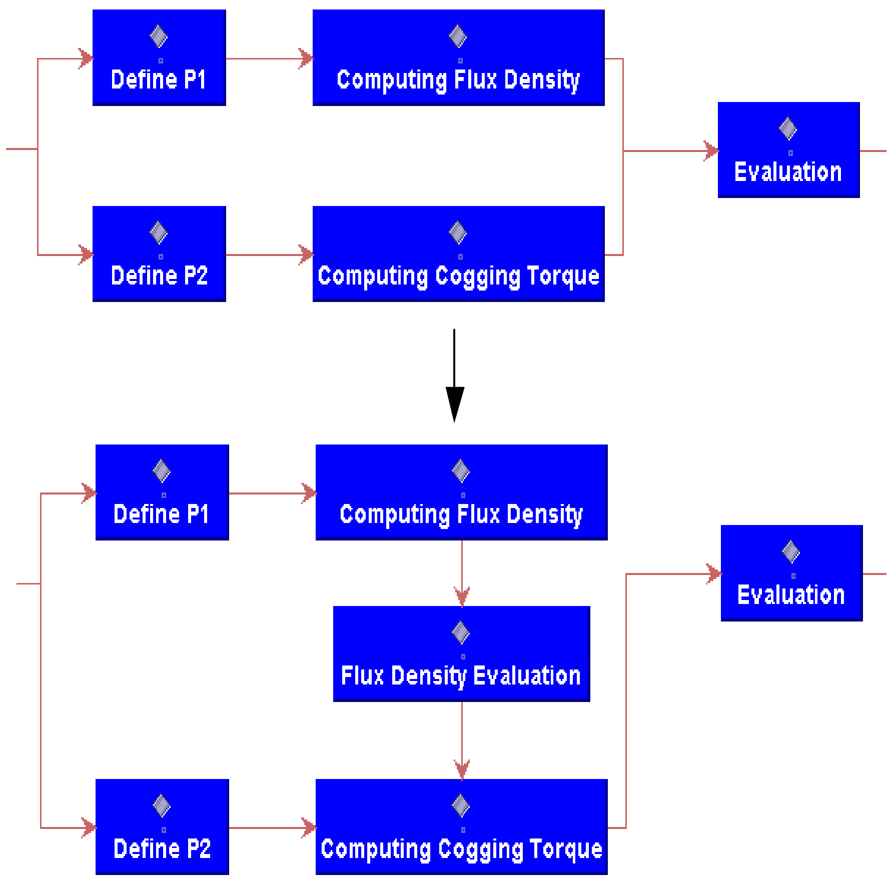


Figure 5.14: Dynamically insert new task into workflow model

5.4 Summary

The CoCADE framework provides a platform for designers from different companies and institutes to communicate their ideas in design stage. In the case study, designers from research institute can provide the technical feasibility analysis for product development. Designers from manufacturing industry can provide advices from the viewpoint of manufacturing process. All these designers follow the predefined workflow process. Collaborative sessions including asynchronous collaborative session and synchronous collaborative session are defined by workflow model to facilitate the collaboration activities. It is shown that the CoCADE framework can effectively support such product development activities.

Chapter 6

Conclusions

6.1 Concluding Remarks on Present Work

Although collaborative design has been intensively studied by researchers in recent years and frameworks on the basis of different technologies have been explored, the effective integration of CAE capabilities to handle multi-physical problems for product performance evaluation remains to be studied and realized. Collaborative session management in distributed engineering design and analysis environment become more complicated because of the involvement of product performance evaluation activities.

This thesis introduces a framework develped on the basis of Microsoft .NET technology and it can provide a collaborative design and analysis environment over the Internet. It appears that the state-of-the-art .NET technology is an effective tools to build such framework. The thesis also discusses the product design process and shows how to leverage workflow technology to organize collaborative sessions. The synchronization problem in synchronous collaboration process has been solved using a new synchronization scheme.

During the course of this research work, several contributions have been achieved:

- A prototype of a distributed collaborative CAD/CAE system, which is based on a client-server architecture comprising of modeling client, coordination server, and analysis server, has been built. The architecture intents to provide reliable support for both design and analysis activities.
- A systematic and comprehensive definition of collaborative session has been given. The architectural structure of collaborative session has been presented, its key functions are described, and issues related to collaborative session management are studied.
- A product design workflow system has been incorporated into the engineering design and analysis environment. Product design process in such prototype system has been analyzed. The mechanism that makes use of workflow to manage collaborative sessions has been illustrated. It is found that the workflow engine can be utilized to dynamically define collaborative sessions during product design process to facilitate product design activities.
- A new multi-thread request response scheme has been developed to facilitate synchronization of initial representation data, operation and session state in collaborative sessions.

Through the case study, it can be concluded that the workflow-drive mechanism can effectively organize collaborative sessions during product design process and the synchronization scheme can efficiently solve synchronization problems in a collaborative session. The proposed framework provides reliable support to CAD and CAE sessions participated by designers from geographically dispersed locations.

6.2 Suggestion on Possible Future Work

Agent technology provides an extension to collaborative session management in product design process. As defined by Jennings and Wooldridge [87], an agent is a computer system situated in some environments, and that is capable of autonomous action in this environment in order to meet its design objectives. Its properties of being autonomous, collaborative, and intelligent are of high interest to researchers in the area of distributed and collaborative design and analysis. The benefits of applying agent technology in proposed framework include resources management.

In case of workflow change, multiple collaborative sessions may have conflicting interests in using some resources, such as product model, CAE solvers, human resource etc. Agent can help to organize these resources to avoid such conflicts.

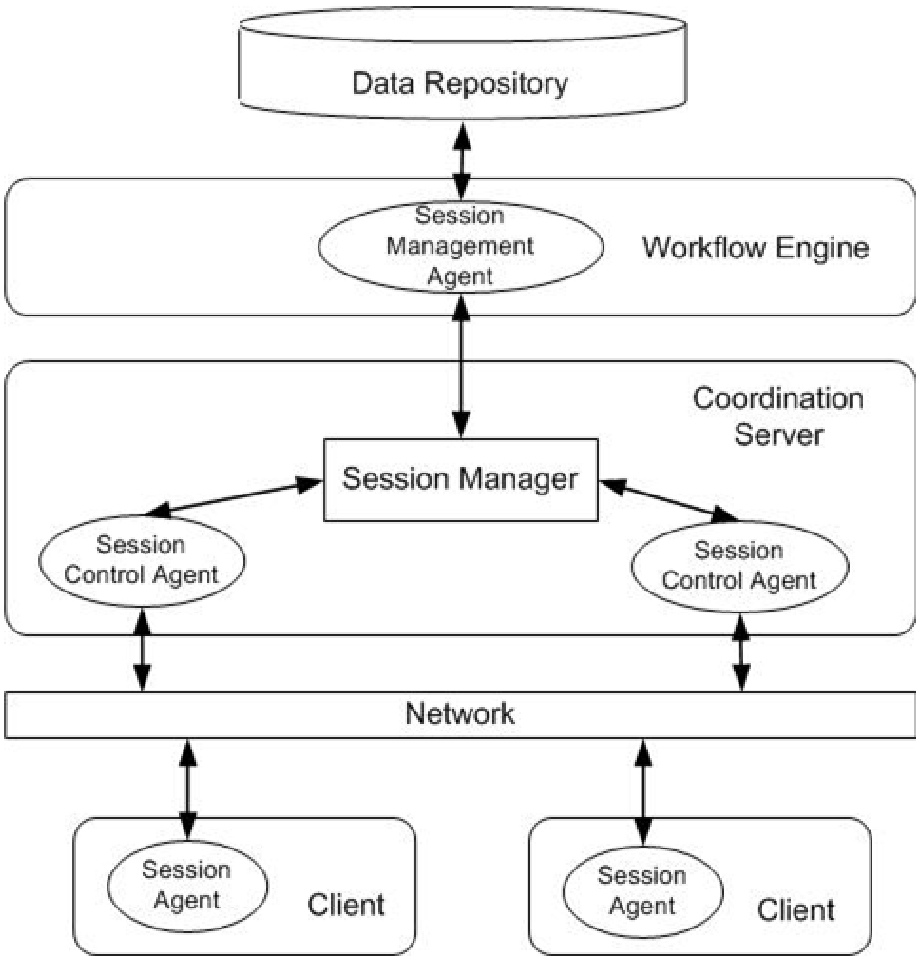


Figure 6.1: Agent enhanced framework

Figure 6.1 shows the extended framework integrated with agents. The main functions of these agents are described as follows.

- **Session agent:** It acts as assistant to client. The typical usage is querying/updating clients' status, sending/receiving message etc.

- **Session control agent:** It manages design process in a session through session agents on the client side, and coordinates with other session control agents for sharing configuration and resources information between sessions.
- **Session management agent:** It helps the workflow engine to manage appropriate resources and other arragement for collaborative sessions according to the workflow model. It will optimize the resources allocation in case of workflow change.

In the extended framework, an agent-enhanced workflow-driven collaborative session management will be realized. The product development activities will be organized in a more flexible, efficient, intelligent and robust manner to facilitate distributed collaborative engineering design and analysis.

Bibliography

[1] AberdeenGroup. Beating the Competition with Collaborative Product Commerce, Jun. 2000.

[2] D. Burdick, “Collaborative Product Commerce: The Technology Vision”, *Research Note Technology by Gartner Group*. Jan. 2000.

[3] M.J. Chung, H.S. Jung, W. Kim, R. Goplannalan, and H. Kim, “A Framework for Collaborative Product Commerce using Web Services”, *Proceedings of the IEEE International Conference on Web Services*, 2004.

[4] ADAMS, MSC.Software Corporation, http://www.mscsoftware.com

[5] ANSYS, ANSYS Inc., http://www.ansys.com/

[6] L. Wang, W. Shen, H, Xie, J. Neelamkavil, and A. Pardasani, “Collaborative conceptual design - state of the art and future trends”, *Computer-Aided Design*, vol.34, pp.981-996, 2002.

[7] Hans-Peter. Dommel and J.J. Garcia-Luna-Aceves, “GROUP COORDINATION SUPPORT for Synchronous Internet Collaboration”, *IEEE INTERNET COMPUTING*, Mar.-Apr. 1999.

[8] CORBA, Object Management Group (OMG) Inc., http://www.omg.org, 2004.

[9] Java/RMI, Sun Microsystems, Inc., http://java.sun.com/products/jdk/rmi/index.html, 2004.

[10] DCOM, Microsoft Corporation, http://www.microsoft.com/com, 2004.

[11] .NET, Microsoft Corporation, http://www.microsoft.com/net, 2004.

[12] N. Senin, N. Borland and D. R. Wallace, "Distributed modeling of product design problems in a collaborative design environment", *Technical Report. CAD lab., Department of Mechanical Engineering, MIT.* 1998.

[13] W. D. Li, S. K. Ong, J. Y. H. Fuh, Y. S. Wong, Y. Q. Lu and A. Y. C. Nee, "Feature-based design in a distributed and collaborative environment", *Computer-Aided Design*, vol.36, pp.775-797, August 2004.

[14] ConceptWorks, RealityWave Inc., http://www.realitywave.com/products-concept.asp, 2004.

[15] OneSpace, CoCreate, http://www.onespace.net, 2004.

[16] eDrawings, SolidWorks Inc., http://www.solidworks.com/edrawings, 2004.

[17] Centric Innovation Center, Centric Software, Inc., http://www.centricsoftware.com/innovate, 2004.

[18] Hoops Streaming Toolkit, Tech Soft America (TSA), http://www.hoops3d.com/products/hoops/stream.htm, 2004.

[19] Autovue, Cimmetry Systems Inc., http://www.cimmetry.com, 2004.

[20] Streamline, Autodesk Inc., http://www.autodesk.com/streamline-trial, 2004.

[21] CollabCAD, National Informatics Centre, India, http://www.collabcad.com, 2004.

[22] Alibre Design, Alibre Inc., http://www.alibre.com, 2004.

[23] I. Greif, "Computer Supported Cooperative Work: A Book of Reasdings", Morgan Kaufmann, San Mateo, Ca, 1988.

[24] L.F. Ludwig, "Integration of CAD/CAE with multimedia teleconferencing and messaging via broadband networks and shared resource servers", *Proceedings*

of the First International Conference on Systems Integration, pp. 136-143, Apr. 1990.

[25] L. Shu, and W. Flowers, "Groupware experiences in three-dimensional computeraided design", *Proceedings of the Conference on Computer-Supported Cooperative Work (1992)*, pp. 179-186, 1992.

[26] M.A. Gisi and C. Sacchi, "Co-CAD: a collaborative mechanical CAD system", *Presence*, vol. 3, pp. 341-350, 1994.

[27] J.P. Harrison, and B. Christensen, "Virtual Collaborative Simulation Environment for Integrated Product and Process Development", *Proceedings of HPDC*, 1996.

[28] M.L. Maher, S.J. Simoff, and A. Cicognani, "Potentials and limitations of virtual design studios", *Interactive Construction On-line*, vol.1, 1997.

[29] A. Stork and U. Jasnoch, "A collaborative engineering environment", *Proceedings of TeamCAD'97 Workshop on Collaborative Design*, pp. 25-33, 1997.

[30] G.D.F. Pahng, N. Senin, and D. Wallace, "Distributed modeling and evaluation of product design problems", *Computer Aided Design*, 30(6):411-423.

[31] J. Y. Lee, H. Kim, S.B. Han, and S.B. Park, "Network-centric feature-based modeling", *Proceedings of Pacific Graphics '99, IEEE Computer Society*, pp.280-289, 1999.

[32] S. Chan, M. Wong and V. Ng, "Collaborative solid modeling on the WWW", *Proceedings of the 1999 ACM Symposium on Applied Computing*, pp.598-602, 1999.

[33] X.D. Liu, "CFACA: component framework for feature-based design and process planning", *Computer-Aided Design*, 32(7):397-408, 2000.

[34] Z. Xie, Z.J. Liu, T.C. Chong, and H. Zhou. "A Framework for Collaborative Engineering Design and Analysis", *The Fourteenth International Conference on the Computation of Electromagnetic Fields*, 2003.

[35] C.A.M. Barbosa, M. Dreux, J. Bento, B. Feijo, R. Melo, and S. Scheer, "An object model for collaborative CAD environments", *The Seventh International Conference on Computer Supported Cooperative Work in Design*, pp. 179-184, 2002.

[36] R. Bidarra, et al., "Web-based Collaborative Feature Modeling", *Proceedings of Sixth ACM Symposium on Solid modeling and applications*, pp.319-320, May. 2001.

[37] W. Shen and D.H. Norrie, "Multi-agent systems for concurrent intelligent design and manufacturing", London, UK: Taylor and Francis, 2000.

[38] M.J. Hague and A. Taleb-Bendiab, "Tool for the management of concurrent conceptual engineering design", *Concurrent Engineering: Research and Applications*, 6(2):111-129, 1998.

[39] G.Q. Huang and K.L. Mak, "Web-based morphological charts for concept design in collaborative product development", *Intelligent Manufacturing*, 10:267-278, 1999.

[40] H.C. Chang, W.F. Lu, and X.F. Liu, "WWW-based collaborative system for integrated design and manufacturing", *Concurrent Engineering: Research and Applications*, 7(4):319-314, 1999.

[41] N. Shyamsundar and R. Gadh, "Internet-based collaborative product design with assembly features and virtual design spaces", *Computer-Aided Design*, 33(9):637-651, 2001.

[42] K.Y. Kim, Y. Wang, O.S. Muogboh, and B.O. Nnaji, "Design formalism for collaborative assembly design", *Computer-Aided Design*, 36(9): 849-871, 2004.

[43] W.D. Li, J.Y.H. Fuh, and Y.S. Wong, "An Internet-enabled integrated system for co-design and concurrent engineering", *Computers in Industry*, in press, May, 2004.

[44] G.D.F. Pahng, S. Bae, and D. Wallace, "A Web-based collaborative design modelling environment", *Proceedings of the IEEE Workshops on Eanbling Technologies Infrastructure for Collaborative Enterprises*, pp.161-167, 1998.

[45] U. Roy and S.S. Kodkani, "Product modelling within the framework of the Would Wide Web", *IIE Transactions*, 31(7):667-677, 1999.

[46] Y.M. Chen, M.W. Liang. "Design and implementation of a collaborative engineering information system for allied concurrent engineering", International Jounal of Computer Integrated Manufacturing, 13(1):11-30, 2000.

[47] J.Y. Lee, H. Kim, and K. Kim, "A web-enabled approach to feature-based modeling in a distributed and collaborative design environment", *Concurrent Engineering*, 9(1):74-87, April. 2001.

[48] S. Nidamarthi, R.H. Allen, R.D. Sriram, "Observations from supplementing the traditional design process via Internet-based collaboration tool", *International Journal of Computer Integrated Manufacturing*, 14(1):95-107, 2001.

[49] S.H. Kong, S.D. Noh, Y.G. Han, G. Kim, and K.I. Lee, "Internet-based collaborative system: press-die design process for automobile manufacturer", *International Journal of Advanced manufacturing Technology*, 20(9):701-708, 2002.

[50] X.G. Ming, Q.F. Ni, W.F. Lu, I.B.H. Lee, M.W. Fu, S.K. Ong, and A.Y.C. Nee, "Towards collaboratively integrated manufacturing system for tooling production in SMEs - the INPROSE approach", *Proceedings of the 9th ISPE International Conference on Concurrent Engineering: Research and Applications*, pp.465-474, 2002.

[51] K. Z. Huang, X. D. Li, S. K. Cao, B. Yang and W. Pan, "Co-DARFAD - the collaborative mechanical product design system", *The Sixth International*

Conference on Computer Supported Cooperative Work in Design, pp.163-168, Jul. 2001.

[52] Y. Gardan, E. Perrin, F. Danesi, L. Denis, N. Gardan, F. Heschung, E. Malik, M. Reimeringer and R. Stock, "First operational systems based on the dija project", *IASTED International Conference on Applied Modelling and Simulation (AMS 2002)*, pp. 294-299, 2002.

[53] L. Denis, Y. Gardan and E. Perrin, "A framework for a distributed CAD system", *Computer Aided Design*, in press, Sep. 2003.

[54] L. Chen, Z.J. Song, and L. Feng, "Internet-enabled real-time collaborative assembly modeling via an e-Assembly system: status and promise", *Computer-Aided Design*, 36(9):835-847, Aug. 2004.

[55] T.J. Nam, and D.K. Wright, "CollIDE: a shared 3D workspace for CAD", *Proceedings of the 1998 Conference on Network Entities*, pp. 389-400, 1998.

[56] T.J. Nam, and D. Wright, "The development and evaluation of Syco3D: a real-time collaborative 3D CAD system", *Design Studies*, 22(6): 557-582, Nov. 2001.

[57] L. Qiang, Y.F. Zhang, and A.Y.C. Nee, "A distributed and collaborative concurrent product design system through the WWW/Internet", *International Journal of Advanced Manufacturing Technology*, 17(5):315-322, 2001.

[58] ACIS R12, Spatial Corporation, http://www.spatial.com, 2003.

[59] Autodesk, Inc., http://www.autocad.com, 2004.

[60] IRONCAD, http://www.ironcad.com, 2004.

[61] Workflow Management Coalition, http://www.wfmc.org.

[62] Hoops 9.0, Tech Soft America (TSA), http://developer.hoops3d.com, 2003.

[63] G. Pahl, and W. Beitz, “Engineering Design”, Springer-Verlag, The Design Council/Berlin, 1984.

[64] V. Hubka, and W. E. Eder, “Theory of Technical Systems - A Total Concept Theory for Engineering design”, ISBN 3-540-17451-6, Springer-Verlag, 1988.

[65] J. S. Jero, and T. McNeill, “An approach to the analysis of design protocols”, *Design Study*, 19:21-61, 1998.

[66] T. Kvan, A. Vera, and R. West, “Expert and situated actions in collaborative design”, in: P. Siriruchatapong, Z. Lin, J. P. Barthes(Eds.), *Proc. of 2nd Intern. Workshop on CSCW in design, International Academic Publishers*, Beijing, pp.400-450, 1997.

[67] B. Prasad, “Concurrent Engineering Fundamentals: Integrated Product and Process Organization”, vol.1, Prentice Hall, 1996.

[68] R. S. Breuhaus, K. R. Fowler, and J. J. Zanatta, “Innovative Aspects of the Boeing 777 Development Program”, *Proc. ICAS*, ICAS96-0.4, vol.1, 1996.

[69] OMG, “Unified Modeling Language (UML), version 1.5”, http://www.omg.org/technology/documents/formal/uml.htm, 2003

[70] Workflow Management Coalition, “Workflow Managment Coalition Terminology & Glossary”, Jun. 1996.

[71] G. Booch, J. Rumbaugh, and I. Jacobson, “The Unified Modeling Language User Guide”, Addison-Wesley Longman, Inc., 1998.

[72] W. Eversheim, H. Rozenfeld, W. Bochtler and R. Graessler, “A Method for an Integrated Design and Process Planning based on a Concurrent Engineering Reference Model,” *Annals of the CIRP* 44, pp.403-406, 1995.

[73] H. Scott, and K. Stephen, “Data Security for Web-based CAD”, *Proc. of the 35th annual conference on Design automation*, vol.00, May. 1998.

[74] F. Danesi, L. Denis, Y. Gardan, Y. Lanuel, and E. Perrin, "Towards a web-based cad system", *Proc. of the 15th International Conference on Information Visualization, Computer Aided Geometric Design Session (IV-2001), 2001, 269. IEEE Computer Society;* pp.269-274, 2001.

[75] S. Bhola., G. Banavar., and M. Ahamad, "Responsiveness and Consistency Tradeoffs in Interactive Groupware," *presented at CSCW*, 1998.

[76] Iometer, http://www.iometer.org.

[77] D. Bertsekas and R. Gallager, "THE M/M/1 QUEUEING SYSTEM" in "Data Networks", 2nd ed. PRENTICE HALL Inc. pp.162-173, 1992.

[78] D. Bertsekas and R. Gallager, "QUEUEING MODELS-LITTLE'S THEOREM" in "Data Networks", 2nd ed. PRENTICE HALL Inc. pp.152-157, 1992.

[79] K. T. Ulrich, and O. Eppinger, "Product Design and Development", *McGraw-Hill International Editions, Management and Organization Series*, 1995.

[80] AMRResearch, http://www.amrresearch.com, 2002.

[81] J. Brown, "The PLM program, an incremental approach to the strategic value of PLM", http://www.technologyevaluation.com, 2002

[82] CIMdata, http:www.cimdata.com, 2002.

[83] X. G. Ming, W. F. Lu, S. Ma, and Q. F. Ni, "Web Service Architecture for Collaborative Product Lifecycle Management in Virtual Enterprise", *The 10th ISPE International Conference on Concurrent Engineering: Research and Applications*, 2003.

[84] R. Menon, H. T. Loh, Z. J. Liu, and Yaccob Ibrahim, "Robust Design of Spindle motors: A case study", *Reliability Engineering and System Safety Quality*, vol. 75, pp.313-319, 2002.

[85] X. K. Gao, T. S. Low, S. X. Chen, and Z. J. Liu, "Robust design for torque optimization using Response Surface Methodology", *IEEE Trans. on Magnetics*, vol.38, No. 2, pp.1141-1144, 2002.

[86] L. Horvath, and T. Varga, "Product Modeling Methods in Collaborative Engineering Environments". *IEEE International Conference on Intelligent Engineering Systems*, pp.517 - 522. 1997.

[87] N.R. Jennings and M.J. Wooldridge, "Applications of Intelligent Agents", In N.R Jennings, M.J. Wooldridge, (eds.), Agent Technology: Foundations, Applications, and Markets. Springer, pp.3-28, 1998.

Appendix A

List of Publications

Recent publications are listed below:

[1] **D.W. Sun**, X.H. Xiong, Z.J. Liu, J.M. Zhao, W.F. Lu, and X. G. Ming, "Concurrency in a Distributed Collaborative CAD/CAE Environment", in press, International Journal of Production Research, Dec. 2004.

[2] **D.W. Sun**, X.H. Xiong, L.W. Ruan, Z.J. Liu, J.M. Zhao, and Y.S. Wong, "Workflow-driven Collaborative Session Management in Product Lifecycle Management via Internet", Proceedings of International Engineering Management Conference 2004, #IEMC245, Oct. 2004.

[3] **D.W. Sun**, L.W.Ruan, Z.J. Liu, J.M. Zhao, W.F. Lu, and X.G. Ming, "Concurrency in a Distributed Collaborative CAD/CAE Environment", Proceedings of the 11th ISPE International Conference on Concurrent Engineering: Research and Applications, #CE04-582, Jul. 2004.

[4] L.W. Ruan, **D.W. Sun**, H.H. Long, and Z.J. Liu, "DATA STREAMING FOR REAL-TIME COLLABORATIVE DESIGN", Proceedings of the International Conference on Scientific and Engineering Computation (IC-SEC) 2004, #IC-SEC0135, Jul. 2004.

Appendix B

List of Abbreviations

Table B.1: List of abbreviations

API	Application Programming Interface
CCM	Centralized Coordination Mechanism
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CoCADE	Collaborative CAD/CAE System
CollIDE	Collaborative Industrial Design Environment
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPC	Collaborative Product Commerce
CPD	Collaborative Product Development
CSCW	Computer Supported Cooperative Work
DCOM	Distributed Component Object Model
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning
FEA	Finite Element Analysis
HSF	Hoops Stream Format
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
J2EE	Java 2 Enterprise Edition
Java/RMI	Java/Remote Method Invocation
LAN	Local Area Network
LODs	Levels of Details
MTRR	Multi-thread Request Response
MTS	Microsoft Transaction Server
NURBS	non-uniform rational B-splines
OMG	Object Management Group
ORB	Object Request Broker
PDM	Product Data Management
PLM	Product Lifecycle Management
RPC	Remote Procedure Call
SDK	Software Development Kit
STEP	Standard for the Exchange of Product model data
SOAP	Simple Object Access Protocol
Syco3D	Synchronous collaborative 3D CAD system
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modelling Language
VRML	Virtual Reality Modeling Language
WfMc	Workflow Management Coalition
WfMS	Workflow management system
WPDSS	Web product design support system
XML	eXtensible Markup Language

Appendix C

Main Visual C# Codes

- **Main Visual C# Codes for Component - Session Manager**

```
namespace CoServer {
    public class SessionManager:MarshalByRefObject,ISessionManager
    {
        //following is the source code for create and join session
        public int CreateSession(string sessionName,string sessionPwd
        ,string clientName)
        {
            //check if the sessionName has been used
            int i=0;
            for (i=0;i<Server.sessionListSize;i++)
            {
                if (Server.sessionList[i].sessionState==1 &&
                Server.sessionList[i].name==sessionName)
                    return -1;
            }
            //find vacant session position
            i=0;
            while (Server.sessionList[i].sessionState!=0 &&
            i<Server.sessionListSize-1)
            {
                i++;
            }
            if (Server.sessionList[i].sessionState!=0)
            {
                Server.e.MaxSessionReached();
                return -10;
            }
            //intialize session
            Server.sessionList[i].sessionState=1;
            Server.sessionList[i].clientSize=1;
            Server.sessionList[i].id=i;
            Server.sessionList[i].name=sessionName;
            Server.sessionList[i].passWord=sessionPwd;
            string nullString="";
            Server.sessionList[i].data[0]=nullString.ToCharArray();
            Server.sessionList[i].informationSize=0;
            for (int index=0; index<Server.informationHistoryLength;
            index++)
            {
                Server.sessionList[i].displayInformation[index]="";
            }
            Console.WriteLine("Infor: Client-{0}-create session-{1}--{2}",
                clientName,sessionName,System.DateTime.Now.ToString());
            //add to clientlist
            Server.clientList[i][0].id=0;
            Server.clientList[i][0].name=clientName;
            Server.clientList[i][0].status="Initiator";
            //return sessionID
            return i;
        }
        public string JoinSession(string sessionName,string
        sessionPwd,string clientName)
        {
            string idString="";
            int sessionID=0,clientID;
            while (Server.sessionList[sessionID].name!=sessionName
            && sessionID<Server.sessionListSize-1)
            {
                sessionID++;
            }
            if (Server.sessionList[sessionID].name!=sessionName)
            {
                //incorrect session name
                idString="-3";
                return idString;
            }
            if (Server.sessionList[sessionID].passWord!=sessionPwd)
                //incorrect pwd
                return idString;
            if (Server.sessionList[sessionID].clientSize==
            Server.clientListSize-1)
            {
                //maximus clients reached
                idString="-2";
                return idString;
            }
            for (int i=0;i<Server.sessionList[sessionID].clientSize;i++)
            {
                if (Server.clientList[sessionID][i].name==clientName)
                {
                    //duplicate client name
                    idString="-1";
                    return idString;
                }
            }
            clientID=Server.sessionList[sessionID].clientSize;
            Server.clientList[sessionID][clientID]=new DataStructure.Client();
            Server.clientList[sessionID][clientID].id=clientID;
            Server.clientList[sessionID][clientID].name=clientName;
            Server.clientList[sessionID][clientID].status="  ----  ";
            Console.WriteLine("Infor: Client-{0}-join session-{1}--{2}",
                clientName,sessionName,System.DateTime.Now.ToString());
            Server.sessionList[sessionID].clientSize++;
            //generate information
            int clientSize=Server.sessionList[sessionID].clientSize;
            Server.clientList[sessionID][clientSize].id=clientID;
            Server.clientList[sessionID][clientSize].name=clientName;
            Server.clientList[sessionID][clientSize].status="Join";
```

```
            //raise event
            for (int i=0;i<clientID;i++)
            {
                Server.sessionStateUpdate[sessionID][i].Set();
            }
            idString=sessionID.ToString()+","+clientID.ToString();
            return idString;
        }
}
```

- **Main Visual C# Codes for Component - Message Handler**

```
namespace CoServer {
    public class MessageHandler:MarshalByRefObject,IMessageHandler
    {
    //following is the source code for update data and token management
        public void UpdateData(int sessionID,char[] newData,int dataSize)
        {
            //data[0] store the number of data string
            Server.sessionList[sessionID].data[0]=dataSize.ToString().ToCharArray();
            //store the data to data[dataSize]
            Server.sessionList[sessionID].data[dataSize]=newData;
            return;
        }
        public char[] GetNewData(int sessionID,int clientID,int sequence,
        bool firstTime)
        {
            if (!firstTime)
                Server.dataUpdate[sessionID][clientID].WaitOne();
            int clientSize=Server.sessionList[sessionID].clientSize;
            string nullChar="";
            if (Server.sessionList[sessionID].sessionState==0)
            {
                return nullChar.ToCharArray();
            }
            else if (Server.clientList[sessionID][clientSize].status[0]=='L')
            {
                if (Server.clientList[sessionID][clientID].dataInform)
                {
                    //if the leave client's id is prior to the current client
                    if (Server.clientList[sessionID][clientSize].id<clientID)
                    {
                        nullChar="0";
                        Server.clientList[sessionID][clientID].dataInform=false;
                        return nullChar.ToCharArray();
                    }
                        //if the leave client is the current client
                    else if (Server.clientList[sessionID][clientSize].id==clientID)
                    {
                        Server.clientList[sessionID][clientID].dataInform=false;
                        return nullChar.ToCharArray();
                    }
                }
            }
            return Server.sessionList[sessionID].data[sequence];
        }
        public void RequestToken(int sessionID,int clientID)
        {
            int clientSize=Server.sessionList[sessionID].clientSize;
            //generate information
            Server.clientList[sessionID][clientSize].id=clientID;
            Server.clientList[sessionID][clientSize].name=
                Server.clientList[sessionID][clientID].name;
            Server.clientList[sessionID][clientSize].status="Requesting";
            //update clientlist
            Server.clientList[sessionID][clientID].status="Requesting";
            //raise event
            for (int i=0;i<clientSize;i++)
            {
                Server.sessionStateUpdate[sessionID][i].Set();
            }
            return;
        }
        public void ReleaseToken(int sessionID,int clientID)
        {
            int clientSize=Server.sessionList[sessionID].clientSize;
            Server.clientList[sessionID][clientSize].id=clientID;
            Server.clientList[sessionID][clientSize].name=
                Server.clientList[sessionID][clientID].name;
            Server.clientList[sessionID][0].status="Initiator(0)";
                Server.clientList[sessionID][clientSize].status="  ----  ";
                Server.clientList[sessionID][clientID].status="  ----  ";
            //raise event
            for (int i=0;i<clientSize;i++)
            {
                Server.sessionStateUpdate[sessionID][i].Set();
            }
            return;
        }
        public void ConferToken(int sessionID,int clientID)
        {
            int clientSize=Server.sessionList[sessionID].clientSize;
            //generate information
            Server.clientList[sessionID][clientSize].id=clientID;
            Server.clientList[sessionID][clientSize].name=
                Server.clientList[sessionID][clientID].name;
            Server.clientList[sessionID][clientSize].status="Operating";
            //update client list
            if (clientID==0)
                Server.clientList[sessionID][clientID].status="Initiator(0)";
            else
                Server.clientList[sessionID][clientID].status="Operating";
            //raise event
            for (int i=0;i<clientSize;i++)
            {
                //update client list
                if (i!=0 && i!=clientID)
                    Server.clientList[sessionID][i].status="  ----  ";
                //raise event
                Server.sessionStateUpdate[sessionID][i].Set();
            }
            return;
        }
}
```

Appendix D

Main Visual C++ Codes

- **Thread for Requesting New Operation**

```
UINT WatchNewDisplay(LPVOID param) {
    bool keepRunning=true;
    CoCADEView* thisView=static_cast<CoCADEView*>(param);
    int theCID=thisView->m_pClientDlgBar->m_pNetInfor->clientID;
    int startPoint=0;
    while (keepRunning)
    {
        //invoice get new data method
        try
        {
            CString displayInfor,typeString,messageString;
            displayInfor=thisView->m_pClientDlgBar->GetDisplayInformation
            (theCID,startPoint);
            startPoint++;
            if (displayInfor=="")
            {
                keepRunning=false;
            }
            else if (displayInfor=="0")
            {
                theCID--;
            }
            else// if (!thisView->m_pClientDlgBar->m_hasToken)
            {
                int iEnd,iLength;
                iEnd=displayInfor.FindOneOf(" ");
                typeString=displayInfor.Mid(0,iEnd);
                iLength=displayInfor.GetLength();
                messageString=displayInfor.Right(iLength-iEnd-1);
                if (typeString=="N_SET_CAMERA")
                {
                    thisView->m_pHView->SetCameraFromMessage(messageString,
                    messageString.GetLength());
                }
                //unpack operations
                ... ...
            }
        }//end of try
        catch (Exception)
        {
            AfxMessageBox("Fail to update display!");
            keepRunning=false;
        }
    }
    return 0;
}
```

- **Thread for Requesting Initial Representation Data**

```
UINT WatchNewData(LPVOID param) {
    bool keepRunning=true;
    CoCADEView* thisView=static_cast<CoCADEView*>(param);
    int theCID=thisView->m_pClientDlgBar->m_pNetInfor->clientID;
    while (keepRunning)
    {
        //invoice get new data method
        try
        {
        if (!thisView->m_pClientDlgBar->GetHSFData(0,false,theCID))
        {
            keepRunning=false;
        }
        else
        {
            //get new data list size
            int new_data_size=atoi(thisView->m_pClientDlgBar->m_pNetInfor->
            hsf_data);
            if (new_data_size==0)
            {
                theCID--;
            }
            else
            {
            if (new_data_size>1000)
            {
                new_data_size=new_data_size % 10000;
                thisView->m_pClientDlgBar->m_pNetInfor->data_size=0;
                ((HCoCADEView*)thisView->m_pHView)->Flush(false);
            }
            //store message length
            int message_length;
            //insert new data
            for (int i=thisView->m_pClientDlgBar->m_pNetInfor->data_size+1;
            i<new_data_size+1;i++)
            {
                thisView->m_pClientDlgBar->GetHSFData(i,true,theCID);
                message_length=
                    thisView->m_pClientDlgBar->m_pNetInfor->hsf_data_length;
                thisView->m_pHView->InsertHSFDataFromMessage
                    (thisView->m_pClientDlgBar->m_pNetInfor->hsf_data,
                    message_length);
            }
            //update view
            thisView->m_pClientDlgBar->m_pNetInfor->data_size=new_data_size;
            thisView->m_pHView->FitWorld();
            thisView->m_pHView->Update();
            }
        }
        }//end of try
        catch (Exception* e)
        {
            AfxMessageBox("Fail to update data!");
            keepRunning=false;
        }
    }
    return 0;
}
```

- **Thread for Requesting Session Status**

```
UINT WatchSessionState(LPVOID param) {
    //set running control
    bool keepRunning=true;
    //get object reference
    ClientDlgBar* pClientDlgBar=static_cast<ClientDlgBar*>(param);
    int theCID=pClientDlgBar->m_pNetInfor->clientID;
    CString theCName=pClientDlgBar->m_pNetInfor->clientName;
    int iend,ilength,clientSize,clientID=0;
    CString stateString,clientName,clientStatus;
    while(keepRunning)
    {
    try
    {
        IMessageHandler* mH=pClientDlgBar->GetMH();
        //get session state, clientStatus,clientName
        ... ...
        stateString=
            mH->GetSessionState(pClientDlgBar->m_pNetInfor->sessionID,
            pClientDlgBar->m_pNetInfor->clientID);
        switch(clientStatus[0])
        {
        case 'J':   //Join
            {
                pClientDlgBar->m_listClients.InsertItem(clientID,clientName,0);
                pClientDlgBar->m_listClients.SetItemText(clientID,1,"  ----  ");
                pClientDlgBar->m_outputMessage.DeleteString(0);
                pClientDlgBar->m_outputMessage.AddString("--------"+clientName+
                " joined");
            }
        break;
        case 'L':   //Leave
            {
                if (theCID!=clientID)
                {
                    if (theCID>clientID)
                    {
                        //update NetInfor
                        pClientDlgBar->m_pNetInfor->clientID--;
                        theCID--;
                    }
                    pClientDlgBar->m_listClients.DeleteItem(clientID);
                    pClientDlgBar->m_outputMessage.DeleteString(0);
                    pClientDlgBar->m_selectedClient="";
                    pClientDlgBar->m_selectedItem=-1;
                    pClientDlgBar->m_outputMessage.AddString("--------"+
                    clientName+" left");
                    if (clientStatus=="LO" || clientStatus=="LKO")
                    {
                        pClientDlgBar->m_listClients.SetItemText(0,1,
                        "Initiator(O)");
                        if (theCID==0)
                        {
                            pClientDlgBar->m_hasToken=true;
                            pClientDlgBar->SetDlgItemText(IDC_BUTTON_SESSIONCONTROL,
                             "Confer Control");
                        }
                    }
                    pClientDlgBar->UpdateClientDlgBar((HWND)param);
                }
                else
                {
                    keepRunning=false;
                    if (clientStatus=="LK" || clientStatus=="LKO")
                    {
                        pClientDlgBar->m_listClients.DeleteAllItems();
                        pClientDlgBar->m_hasToken=false;
                        pClientDlgBar->m_pNetInfor->statusString="Kick";
                        AfxMessageBox("You are drop out of the session.");
                    }
                }
            }
            break;
        case ’T’:   //Terminate
            {
                keepRunning=false;
                if (theCID!=0)
                {
                    pClientDlgBar->m_hasToken=true;
                    pClientDlgBar->m_pNetInfor->statusString="";
                    pClientDlgBar->m_outputMessage.DeleteString(0);
                    pClientDlgBar->m_outputMessage.AddString
                    ("--------Session was terminated");
                    pClientDlgBar->GetClientList(true);
                }
            }
            break;
        case ’ ’:   //release
            {
                if (theCID==0)
                {
                    pClientDlgBar->m_hasToken=true;
                    pClientDlgBar->SetDlgItemText(IDC_BUTTON_SESSIONCONTROL,
                     "Confer Control");
                }
                pClientDlgBar->m_listClients.SetItemText(0,1,"Initiator(O)");
                pClientDlgBar->m_listClients.SetItemText(clientID,1,clientStatus);
                pClientDlgBar->m_outputMessage.DeleteString(0);
                pClientDlgBar->m_outputMessage.AddString("--------"+clientName+
                " released control right");
                pClientDlgBar->m_outputMessage.DeleteString(0);
                pClientDlgBar->m_outputMessage.AddString
                ("--------Initiator obtained control right");
```

```
                pClientDlgBar->UpdateClientDlgBar((HWND) param);
            }
            break;
        case 'O':   //Operating
            {
                for (int i=1;i<pClientDlgBar->m_listClients.GetItemCount();i++)
                    pClientDlgBar->m_listClients.SetItemText(i,1,"  ----  ");
                //update client list to idle
                pClientDlgBar->m_listClients.SetItemText(clientID,1,clientStatus);
                pClientDlgBar->m_listClients.SetItemText(0,1,"Initiator");
                if(theCID==clientID)    //confer to the current client
                {
                    pClientDlgBar->m_hasToken=true;
                    pClientDlgBar->SetDlgItemText
                    (IDC_BUTTON_SESSIONCONTROL,"Release Control");
                    AfxMessageBox("Get control right.");
                }
                pClientDlgBar->m_outputMessage.DeleteString(0);
                pClientDlgBar->m_outputMessage.AddString
                ("--------"+clientName+" was granted control right");
                pClientDlgBar->UpdateClientDlgBar((HWND)param);
            }
            break;
        case 'R':   //Requesting
            {
                pClientDlgBar->m_listClients.SetItemText(clientID,1,clientStatus);
                if (pClientDlgBar->m_hasToken)
                {
                    AfxMessageBox(clientName+
                    " made a request for you to release control.");
                }
            }
            break;
        case 'M':   //Message
            {
                pClientDlgBar->m_outputMessage.DeleteString(0);
                pClientDlgBar->m_outputMessage.AddString(clientName);
                pClientDlgBar->UpdateClientDlgBar((HWND)param);
            }
            break;
        default:    //others -- unknown status
            {
                AfxMessageBox("Unkown session status occured!");
                keepRunning=false;
            }
            break;
        }
    }//end of switch
    catch (Exception* e)
    {
        AfxMessageBox("Fail to update session state!");
        keepRunning=false;
    }
    }
    return 0;
}
```