CADDAC: Multi-Client Collaborative Shape Design System with Server-based Geometry Kernel

Karthik Ramani, Abhishek Agrawal, and Mahendra Babu
School of Mechanical Engineering, Purdue Research and Education Center for Information Systems in Engineering (PRECISE), Purdue University, West Lafayette, IN 47907-1288

Christoph Hoffmann
Department of Computer Sciences, Purdue Research and Education Center for Information Systems in Engineering (PRECISE), Purdue University West Lafayette, IN 47907-1288

New and efficient paradigms for web-based collaborative product design in a global economy will be driven by increased outsourcing, increased competition, and pressures to reduce product development time. We have developed a three-tier (client-server-database) architecture based collaborative shape design system, Computer Aided Distributed Design and Collaboration (CADDAC). CADDAC has a centralized geometry kernel and constraint solver. The server-side provides support for solid modeling, constraint solving operations, data management, and synchronization of clients. The client-side performs real-time creation, modification, and deletion of geometry over the network. In order to keep the clients thin, many computationally intensive operations are performed at the server. Only the graphics rendering pipeline operations are performed at the client-side. A key contribution of this work is a flexible architecture that decouples Application Data (Model), Controllers, Viewers, and Collaboration. This decoupling allows new feature development to be modular and easy to develop and manage. [DOI: 10.1115/1.1582882]

Keywords: Collaborative Product Design, BREP, CAD/CAM, Solid Modeling

Introduction

There have been tremendous changes in product development processes in recent years. A significant increase in the outsourcing of components necessitates increased coordination and collaboration within and across enterprises. Balakrishnan *et al.* [1] define collaboration as the process of interaction among multiple participants with inter-linked but distinct roles and specializations working together on inter-related activities towards a common goal.

Collaborative product design based information systems are slowly becoming a reality due to the advent of the Internet and powerful computers. However, collaboration for product design requires considerably broader capabilities than are provided by most of the other simple collaborative applications available today. The current systems demand high bandwidth, high-end software and hardware resources at the client-side. Consequently, true Internet-based collaborative product design systems are not yet available.

Our research prototype, CADDAC, provides a platform to support collaborative product design over the Internet. The goal of this prototype is to develop the framework of a thin-client collaborative system for shape conceptualization. CADDAC employs a three-tier architecture that partitions the overall system into a server-side, a client-side, and a database. A master copy of the CAD model is kept at the server and each client has a local copy of this master. The clients are capable of creating, modifying, and deleting the master CAD model using the collaboration platform. All of the solid modeling and constraint solving operations are performed at the server-side in order to have thin clients.

The purpose of the paper is to demonstrate the ability of our thin-client architecture to collaborate. Ability to perform the following functions, which are not done by any research prototype or commercial system, demonstrates the uniqueness of the architecture.

1. Creating and modifying geometry in a collaborative mode.
2. Having a thin-client model by performing the computation intensive operations like faceting, solid-modeling operations and constraint solving on the server.
3. Maintaining the shape creation history in the database on the server-side. This information could be retrieved for reviewing when needed.

The primary contribution of our work lies in the architecture that allows development of new feature modules to be seamless plug-ins into the existing architecture. A Client structure is based on a Model/View/Controller (MVC) pattern. In the MVC architecture, the Controllers and Views can be added when additional information is added to the Model. Further, the *Command* Object distribution mechanism responsible for collaboration is independent of the commands being passed. The decoupling of Application Data (Model), Controllers, Viewers, and Collaboration makes new feature additions very easy.

1 Review of Previous Work

There have been some studies on collaborative product design over the Internet. Initial attempts were mainly to support data exchange, collaborative viewing and mark-up, and shared 2D sketching. CollIDE [2] and CyberEye [3] are 3D shared workspaces that can be accessed by multiple users, but they lack the capability to perform collaborative creation, deletion, or the manipulation of geometry.

There has also been some work on collaborative geometric modeling and co-editing over the Internet. Kao [4] proposed a collaborative CAD/CAM system (COCADCAM) that includes surface modeling, tool path simulation and post-processing, and CAD geometry co-editing, but no solid modeling. Collaborative Solid Modeling [5] allows sharing and editing of a solid model over the web synchronously, but requires a solid modeler at each client. WebSPIFF [6,7] is a collaborative feature based modeler that allows simultaneous creation, deletion, and modification of features by many users. Teledesign [8] examines groupware interface issues with collaborative 3D modeling environments. MUG [9] is a multi-user environment for collaborative conceptual shape design capable of modeling NURBS one at a time. NetVP [10] is a distributed design environment for network-centric virtual prototyping. NetVP is capable of geometry creation using features but not using sketching. A survey of the other collaboration systems in literature was presented in our earlier research [11].

There are several commercial systems also available. Webscope™ [12] is a real-time product data communication system that allows multiple users to simultaneously view, annotate, and query CAD models and documents. OneSpace Designer™ [13] allows collaboration data to be viewed, marked-up, edited, and saved, but it cannot create new geometry. DIVISION™ [14] provides support for 3D and 2D product visualization, mock-up, and review. Alibre Design™ [15] allows users to collaboratively perform geometric modeling, but it needs a solid modeling kernel at each client that is capable of doing modeling. Alventive™ [16] has a module for collaborative viewing and markup but it is not

Contributed by the Computer Aided Product Development (CAPD) Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received Dec. 2001; Revised Apr. 2003. Associate Editor: P. Wright.

 Copyright © 2003 by ASME

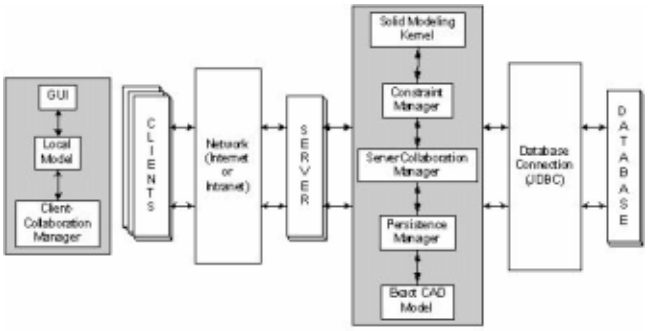

Fig. 1 Three-tier architecture showing client-side, server-side, and database and the interfaces between them

capable of collaborative modeling. Some of the other systems are GS-Design™ [17], IX SpeeD™ [18], and 3G.Web.Decisions™ [19].

2 Overview of the System

This research prototype, CADDAC, provides a web-based and platform-independent system to create, delete, and edit three-dimensional geometry in a collaborative and synchronous mode. Most of the computationally intensive functionalities, such as solid modeling and constraint solving operations, are performed at the server in order to reduce the size of the clients. A core portion of the system is a flexible architecture that can be expanded to include other collaborative features.

The user can apply and modify constraints and dimensions on the geometry. Furthermore, the user can import existing CAD geometries in common file formats such as STEP and IGES for viewing and mark-up purposes. CADDAC also has a user-login system that allows different users to have various privileges. For example, some users have the privilege to make changes in the design while others can only view the model.

2.1 CADDAC Architecture. CADDAC, as shown in Fig. 1, is designed based on three-tiers encompassing a client-side, a server-side, and a database. The server-side contains the master copy of the CAD model, solid modeling kernel, *Constraint Manager, ServerCollaboration Manager*, and *Persistence Manager*. The client-side has a copy of the master CAD model, and this model is displayed to the user through a GUI. The client-side also has a *ClientCollaboration Manager* that keeps a perpetual connection to the server in order to send and receive commands from the server. There is a communication layer between the clients and the server that performs the task of transferring data through a network medium, such as the Internet or an intranet. There is also a layer between the server and the database that provides an interface to save and retrieve data from the server.

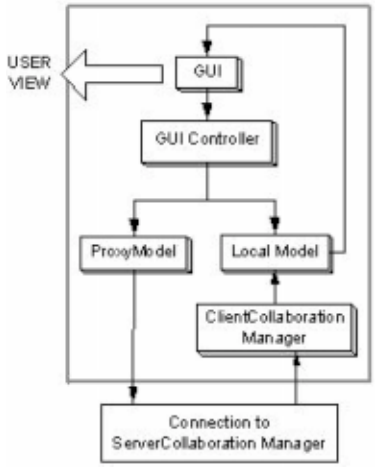

Fig. 3 Structure of MVC pattern based client with connection to the server

2.1.1 CADDAC Server. The CADDAC server provides support for solid modeling, constraint solving operations, data management, session management, and synchronization of clients. The clients have the option of joining an already existing session or creating a new session. The server keeps track of the sessions in progress (Fig. 2).

The server receives data from all the clients and sends the processed data back to the clients. The server also holds the master copy of the CAD model that is used to update the local version of the shape model on the clients. A faceted model with persistent and unique face IDs is generated from the exact master CAD model, using the solid modeling kernel. The faceted model is sent to all of the clients through the *ServerCollaboration Manager*. The *ServerCollaboration Manager* maintains a reference to every client that is connected to the server, and this reference is used to send and receive commands and data from the clients.

Command objects are used to transmit data and instructions between the server and the clients. These objects propagate the changes made by one client to the server and then from the server to other clients. There is a unique *Command* object defined for each method of the local model on the client-side. These method-

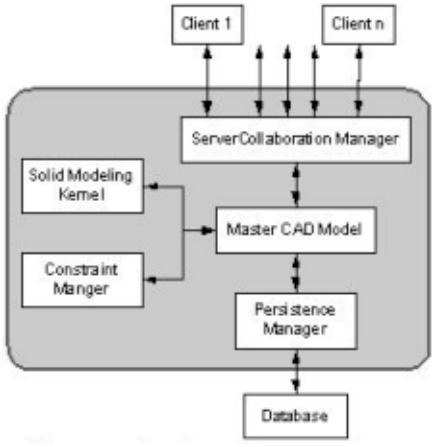

Fig. 2 Server architecture showing various components and the interface to the clients

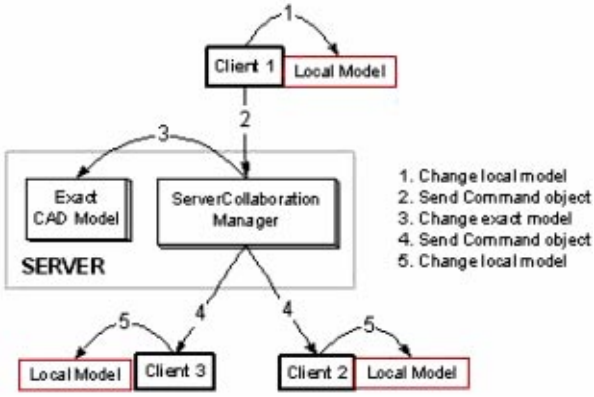

Fig. 4 Synchronization process showing a series of operations starting at one of the clients indicated by arrow 1 followed by 2, 3, 4, and 5

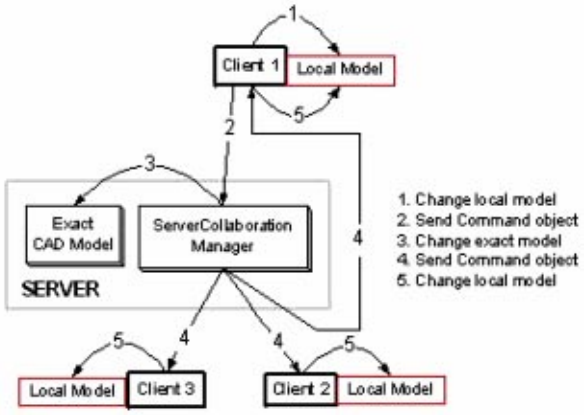

Fig. 5 Synchronization process showing a series of operations starting at one of the clients indicated by arrow 1 followed by 2, 3, 4, and 5, ending the process at client 1

specific *Command* objects store all the arguments required for the corresponding method. When a method to change the local model on the client-side is invoked, a *Command* object for that particular method is created. This object is sent to the server using the communication layer between the server and the clients. Once the object reaches the server, the object calls its corresponding method of the exact model and thus updates the exact model. Broadcasting of Command objects to other clients follows this. Other clients also update their local model in a similar fashion.

The *Constraint Manager* is used to solve two-dimensional constraints imposed on the geometries by the clients. Some examples of the types of constraints available in CADDAC are horizontal, vertical, length, radius, and perpendicularity.

The *Persistence Manager* takes care of storing data in the database. The manager uses JDBC data access APIs to store the Java representation of the sketch(s) and the exact shape model into the database.

The *Solid Modeling Kernel* is needed to create three-dimensional representation of the shape model from the sketch input given by the clients. ACIS™ is used as the solid modeling kernel. ACIS™ provides the software routines to represent three-dimensional solids and to perform operations such as Boolean addition, intersection, or subtraction on these solids.

The history of the shape creation process is also stored in the form of a list of primitives. These primitives store the sketches and the details of solid operations performed on those sketches. This history information allows the user to modify an existing sketch or the solid operation parameters such as an extrusion depth or angle.

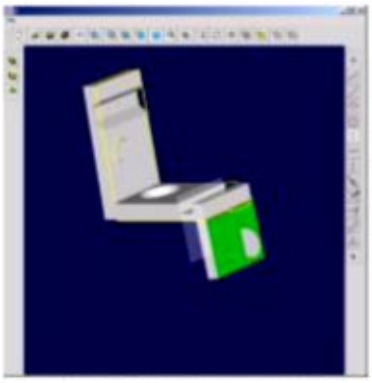

Fig. 6 Part Modeled using CADDAC

Fig. 7 Model imported from commercial software

2.1.2 CADDAC Client. Only the master-client is capable of doing real-time creation, editing, and deletion of geometry over the network without having the solid modeling and constraint solving libraries locally. Other clients are capable of viewing the three-dimensional model and manipulating it by applying rotational, translation, and scaling transformations. These other clients can become the master-client by requesting control from the current master client.

Client structure is based on a Model/View/Controller (MVC) pattern (Fig. 3) [20]. Viewer is implemented using a Java wrapper over opengl. Controller is implemented using the basic toolbar and mouse event handler classes provided as a part of core Java. Model object is the client's local model, which is a faceted representation of the master CAD model with unique and persistent face-ids. Any changes made in the local model through the Controller or the *ClientCollaboration Manager* is immediately notified back to the View. In addition, any modifications made by any of the clients connected to the server are displayed on the GUI of the rest of the clients.

The *ClientCollaboration Manager* takes care of the incoming commands from the server and modifies the local model based on that command. A *ProxyModel* object implements a proxy pattern to substitute for the master model on the server. All the methods of the local model are also implemented in *ProxyModel*, but the methods in *ProxyModel* create a *Command* object that is specific for each method. There are two ways in which the server can respond to an incoming *Command* object. If the *Command* object does not involve any interactions with the solid modeling kernel and the *Constraint Manager* then the *Command* object is directly sent to other clients as shown by arrow 4 in Fig. 4. However, if there is any such interaction involved, e.g., constraint solving, boolean subtract, or a sweep, then a new *Command* is generated at the server which contains the result of these operations. This new *Command* object is then sent to all of the clients as shown by arrow 4 in Fig. 5. Once these objects reach the clients, the local model at that client is updated by calling the method corresponding to the received *Command* object.

3 Results

A solid model with several sketches and solid operations is shown in Fig. 6. A STEP file with 6138 facets imported using our system is shown in Fig. 7.

4 Conclusions

In this paper, a system with web-based collaborative design is presented. One of the characteristic features of the system is that it does not require either a solid modeler or a constraint solver at the client-side. These operations are performed at the server so that the clients are as thin as possible. This leads to easy installation and use of the system by people who have limited hardware and software resources.

In our system, addition of new features translates to information relating to those features being added to the Model class. In MVC architecture, the Controllers and Views can grow as the Model grows. Further, the *Command* Object distribution mechanism responsible for collaboration is independent of the Commands being passed to the clients. The decoupling of Application Data, Controllers, Viewers, and Collaboration makes new feature additions easy. The application data of the new features would be represented in classes, which are made a part of the existing *Model* class. Each of the new features will have its own Controller class, which is plugged into the existing list of Controllers. Similarly, the new features will have their own Viewer classes, which are registered with the Model. Thus, the new features would be seamless plug-ins into the existing architecture.

Although collaborative product design and manufacturing systems are still at their infancy, it is expected that with the developments in network technologies, these systems will play a crucial role in expanding distributed design and manufacturing. We view our prototype as a contribution towards this realization.

Acknowledgments

We acknowledge Bo Yuan for implementing a constraint solver incorporated in CADDAC and Spatial Corporation for providing ACIS™ software for use in Purdue Research and Education Center for Information Systems in Engineering (PRECISE). We also acknowledge the Indiana 21st Century Research and Technology Funds and the National Science Foundation (#0227828) for funding this research.

References

[1] Balakrishnan, A., Kumara, S. R. T., and Sundaresan, S., 1999, ''Manufacturing in the Digital Age: Exploiting Information Technologies for Product Realization,'' Information Systems Frontiers, **1**, pp. 25–50.

[2] Nam, T. J., and Wright, D. K., 1998, ''CollIDE: A Shared 3D Workspace for CAD,'' 1998 Conference on Network Entities, Leeds, UK.

[3] Zhuang, Y., Chen, L., and Venter, R., 2000, ''CyberEye: An Internet-Enabled Environment to Support Collaborative Design,'' Concurrent Engineering: Research and Applications, **8**(3), pp. 213–229.

[4] Kao, Y. C., and Lin, G. C. I., 1998, ''Development of a Collaborative CAD/CAM System,'' Robotics and Computer-Integrated Manufacturing, **14**, pp. 55–68.

[5] Chan, S., Wong, M., and Ng, V., 1999, ''Collaborative Solid Modeling on the WWW,'' Proc. of the 1999 14th ACM Symposium on Applied Computing, San Antonio, Texas, pp. 598–602.

[6] Bidarra, R., Van Den Berg, E., and Bronsvoort, W. F., 2001, ''Interactive Facilities for Collaborative Feature Modeling on the Web,'' Proc. of the Tenth Portuguese Conference on Computer Graphics, Lisbon, Portugal, pp. 43–52.

[7] Bidarra, R., Van Den Berg, E., and Bronsvoort, W. F., 2001, ''Web-based Collaborative Feature Modeling,'' Proc. of Solid Modeling '01-Sixth Symposium on Solid Modeling and Applications, pp. 319–320.

[8] Shu, L., and Flowers, W., 1992, ''Groupware Experience in Three-Dimensional Computer-Aided Design,'' Proceedings of CSCW'92, pp. 179–186.

[9] Foster, C. V., Shapirstein, Y., Cera, C. D., and Regli, W. C., 2001, ''Multi-User Modeling of Nurbs-Based Objects,'' Proceedings of DETC'01, 2001 ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference (DETC2001/CIE-21256), Pittsburgh, Pennsylvania.

[10] Lee, J. Y., 2001, ''Shape Representation and Interoperability for Virtual Prototyping in a Distributed Design Environment,'' Int. J. Adv. Manuf. Techno, **17**, pp. 425–434.

[11] Agrawal, A., Ramani, K., and Hoffmann, C., 2002, ''CADDAC: Multi-Client Collaborative Shape Design System with Server-Based Geometry Kernel,'' 2002 ASME Design Engineering Technical Conference, (DETC2002/CIE-34465), Montreal, Canada, pp. 57.

[12] Webscope Inc., 2001, Webscope™ Software, Sunnyvale, CA. http://www.webscopeinc.com/products/features.html

[13] CoCreate Software Inc., 2001, OneSpace Designer™ Software, Fort Collins, CO. http://www.cocreate.com

[14] PTC, 2001, DIVISON™ Software, Needham, MA. http://www.ptc.com/products/division/index.htm

[15] Alibre Inc., 2001, Alibre Design™ Software, Richardson TX. http://www.alibre.com

[16] Alventive Inc., 2001, Quick Collaboration™ Software, Santa Clara CA. http://www.alventive.com

[17] Collabware Corp., 2001, GS-Design™ Software, Pittsburgh, PA. http://www.collabware.com

[18] ImpactXoft, 2001, IX-SPeeD™ Software, San Jose, CA. http://www.impactxoft.com

[19] 3G Corp., 2001, 3G.web.decisions™ Software, Los Angeles CA. http://www.3gacorp.com

[20] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., 1995, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Upper Saddle River, NJ.