Proceedings of the TMCE 2004, April 12-16, 2004, Lausanne, Switzerland, Edited by Horváth and Xirouchakis

SECURE COLLABORATIVE ASSEMBLY AND ASSEMBLY STREAMING

Mahendra Babu Arugundram Hari Krishna Moorthy
PRECISE and Department of Computer Sciences
Purdue University
USA
marugund@purdue.edu

Karthik Ramani
PRECISE and School of Mechanical Engineering
Purdue University
USA
ramani@purdue.edu

ABSTRACT

Supply chain collaboration using the internet has not progressed with increased outsourcing. Activities in different locations are still sequential, reducing productivity. A secure collaborative product assembly system would help designers working in parallel detect the assembly discrepancies at an early stage, thereby reducing the product development time. This paper presents the architecture of the collaborative assembly system which has been developed using CollabFramework, a flexible software framework for collaborative systems. During the assembly session the CAD files are faceted on the server and only their facet representation is delivered to the client. This eliminates security concerns since the actual CAD file with its embedded intelligence is not transferred to the client machine. The assemblies generated tend to be large in size. To minimize the user waiting period streaming of the assembly components is done. Instead of downloading a complete copy of the facet data, the data is distributed on demand, thus effectively utilizing the network bandwidth. Our streaming strategy combines several techniques, including levels of detail, progressive refinement and view based streaming to deliver only enough data to accurately represent the assembly.

KEYWORDS

Collaborative Product Design, Assembly Streaming

1. INTRODUCTION

Ready access to product data represents a competitive advantage in the current scenario where original equipment manufacturers (OEMs) are out-sourcing engineering activities performed internally (Park, M., et al., 1999). Different components or sub-assemblies of the product are designed by different groups of designers at geographically different locations resulting in potential design conflicts (Figure 1). The ability to access, present, and interact with product data in a visual medium and proceed concurrently at different locations dramatically expedites key decisions. The visualization fundamentally empowers users to analyze and interact with that data, increasing their ability to deliver value to their customers. Rapid, accurate access to in-progress designs and related product information eliminates the miscommunication that leads to bottlenecks, thereby reducing the product development lead-time and manufacturing cost to a large extent (Ullman, D. G., 1997).

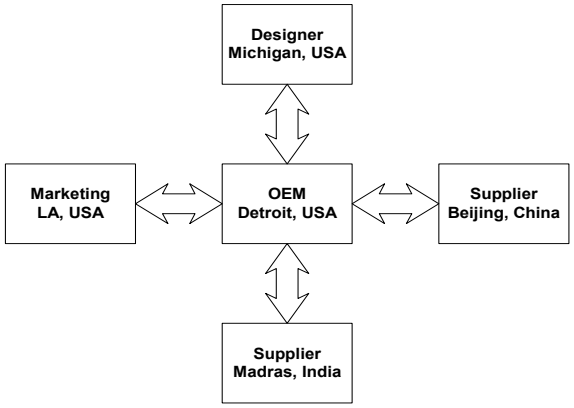

Figure 1 Outsourcing requires concurrent and secure access to information in a global environment

Assembly discrepancies left undetected typically results in a large waste of product development time. Current visualization and collaboration tools help by providing access to non-CAD users, but they are rendered useless by the large quantity of data that has to be delivered. To address this need we have

developed a secure collaborative product assembly module (CAS), which would help designers detect the assembly discrepancies at an early stage. This paper presents,

1. The software architecture of CAS,
2. Streaming strategy that minimizes the user-waiting period streaming by effectively utilizing the network bandwidth, and
3. Security sandbox provided for the assembly subcomponents.

2. RELATED RESEARCH

Cutkosky, M. R., et al. (1996) demonstrate the practicability of collaborative design and validates the Internet as an information infrastructure for collaborative engineering. Regli, W. C., et al., (1997) presents the current practices and future trends in the development of a network based computer aided design framework. Maxfield, J., et al., (1995) discuss an environment that allows real time collaboration between designers for assembly modeling. Roy, U., et al., (1997) present a system where, collaborating designers create concept models of parts, and iterate until the part design is complete prior to manufacturing analysis of the part. Floriani, L. D., et al., (1991) and Requicha, A.G., et al., (1991) present representations for modeling assemblies.

Existing collaborative assembly modeling tools utilize polygonized models or detailed CAD model of the assembly. Polygonized model support only visualization and do not support constraint specification between subcomponents. The boundary-based assembly representation schemes are detailed representations and are therefore not compact enough for transmitting over the limited bandwidth of the Internet. Shyamsundar, N., et al., (2001) and (2002) present a compact internet centric product assembly representation.

Another major restriction of the current assembly systems is that they do not support streaming of the subcomponents. So the user has to wait for the entire data file to be downloaded before accessing or working with the assembly. Autovue by Cimmetry systems (2001) supports streaming of CAD parts. RealityWave's VizStream® (RealityWave, 2001) platform is another commercial product that supports product definition data.

CAS utilizes an intelligent polygonized format that maintains pointers to the corresponding entities in the detailed CAD model of assembly. CAS allows large, complex product definition data to be visualized from anywhere, over any speed network connection. CAS dynamically prioritizes the data stream by using a tight coupling between the clients and the server. The client determines what data is needed. The server delivers the requested data.

3. STREAMING

In traditional assembly systems, applications needed to retrieve an entire data file before the user or receiving application could begin to see, access, or work with the data contained. This slows down work and can frustrate users. Streaming applications partially alleviate this issue by enabling access to the data before the entire file is downloaded (RealityWave, 2001). The key to fast streaming is determining the order in which to send the data. An important measure of the effectiveness of a streaming application is how well the data is ordered in relation to how that data will be used. Figure 2 shows the relationship between time and data usefulness for the streaming and non-streaming scenarios.

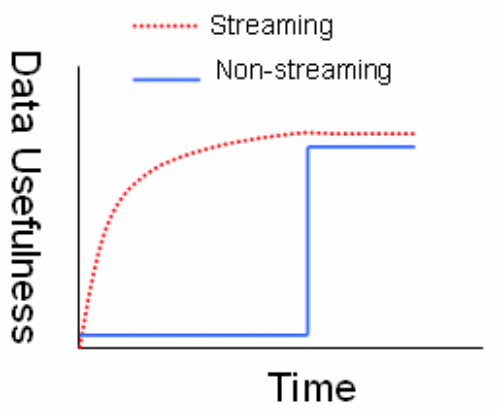

Figure 2 Streaming vs. Non-streaming

3.1. Dynamically Prioritized Streaming (DPS)

Streaming was initially used for video and audio information, for which there is an obvious "beginning-to-end" order. In case of 3D models there is no obvious order in which users will interact. One option for 3D data streaming is to approach data order in a predetermined fashion, much like audio and video streaming. Using general rules, we can predict the order in which the data will be used. For example, a general rule may be to prioritize the data downloaded for a car by the parts of the car. One user

may want to inspect the car's engine, while another is concerned primarily with the body, and yet another wants to see the interior of the car. Predetermined streaming (PS) applications work well when the data sets are small. As 3D model complexity and size increase, predicting the order in which a user will want to work with the data fails.

In order to maximize the effectiveness of streaming, in CAS, data is delivered in an order dependant on the user's interaction. In a truly interactive application, the manner in which the user will interact with the application is not known ahead of time, and therefore the data can not be ordered ahead of time (RealityWave, 2001). CAS determines what the user is doing and presents the information needed accordingly. If the user is examining a car, for example, CAS determines that the user is inspecting the car's hood and downloads more information in that zone; or if the user has moved to the interior, CAS streams the information about the car's interior.

CAS achieves DPS by basing the streaming order on the user's actions and communicating that information between the client and the server. The client determines what the user is viewing, measures the visibility of items in the viewing space, and requests the data in an order that is specific to the user's viewpoint. Since this is a dynamic process, the data is optimally ordered in relation to the user's changing needs.

3.2. Progressive Refinement

The representation of objects is compressed by the following methods that further reduce the amount of information that has to be transmitted.

- Progressive refinement allows a simplified representation to be replaced by more detailed meshes, which improve the visual quality that the user sees over time.
- Physical compression of the polygonized data is used to compress the data that is transmitted. This is done using a variation of the compression and decompression algorithms by Ziv, J., et al (1977).

The usefulness of any visualization application is measured by the speed with which the user can begin interacting with the desired data. DPS facilitates a user's interaction with 3D data, but streaming alone does not necessarily provide the data in a usable form.

One method of streaming would be to separate a 3D assembly into each of its geometrically defined parts and send each part in its entirety in a particular order. This method has a 'popping' effect, as each part comes into view in the order received. To illustrate this with the car example, the system could dynamically determine that the user is focusing on the engine, and then send all of the data for each component of the engine, allowing each component to appear in detail before sending the data for the next component. The cylinder would suddenly appear in its complete form, and then the piston would appear in its complete form, and so on. This process delays data interaction early on because the user can not see all of the parts. They have difficulty understanding where parts are located in relation to other, so they have to wait for more data to arrive before beginning to work.

CAS uses an alternative method called progressive refinement. Progressive refinement creates simplified mesh approximations of each individual part's geometry and downloads them in a hierarchical order, from coarse to fine detail. Each mesh is then successively replaced with a finer level of detail. In other words, a rough image of each component of the engine would appear, then the finer version of the components, and then the entire data of the components. The user quickly sees where the parts are located in relation to others and can immediately begin interacting with the 3D data. As with PS, simplified mesh approximations have limited benefits as the size and complexity of a model increases. The CAS Platform leverages progressive refinement to maximize the speed at which the user can interact with and visualize the requested data.

The ACIS™ (Spatial, 2003) solid modeling kernel is used to obtain the faceted representation of the subcomponents at various levels of detail.

3.3. Culling

CAS uses occlusion and back-face culling to avoid streaming data that the user cannot see because at the user's current viewpoint the data lies behind some other object or outside his view. CAS prepares 3D data to take advantage of culling. As shown in Figure 3 culling reduces the amount of data that has to be streamed thus helping maximize the use of available bandwidth. It also limits how much data needs to be rendered, therefore improving client performance.

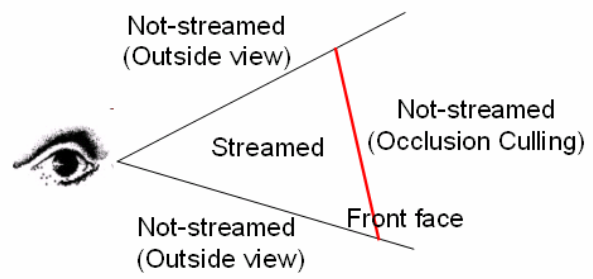

Figure 3 Culling

4. CAD FILE SECURITY

The clients collaborating in a project are provided their own storage locations in the database, which is guaranteed to be secure. The clients upload the CAD files into their respective location in the database. Then during the assembly session they choose the subcomponents forming the assembly from the database.

The local model at client end consists of a list of *Face* objects besides other bookkeeping data, e.g. current display mode, sketch data, selected geometry, and selected face. *Face* object represents a face of the resulting three-dimensional solid. A *Face* consists of an ordered list of the vertices of the face and a list of *Facet* objects. *Facet* list contains the list of triangles generated by faceting the face of the solid. This structure is better than storing only the list of facet triangles as it can show the correct wire frame view of the solid model. This structure also maintains the information of all the triangles belonging to a face that makes it possible to pick a face at the client-side. This faceted representation on the client end is a minimal representation of the CAD file. This guarantees that the sensitive B-rep data in the CAD file is not shared among the other clients thereby ensuring that only the owner of the CAD can access it.

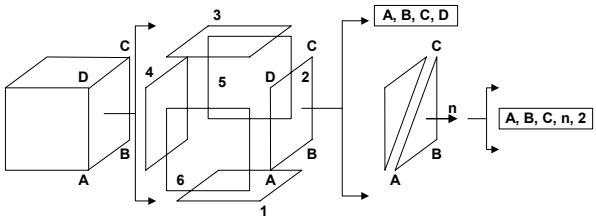

Figure 4 Faceted representation of a cube

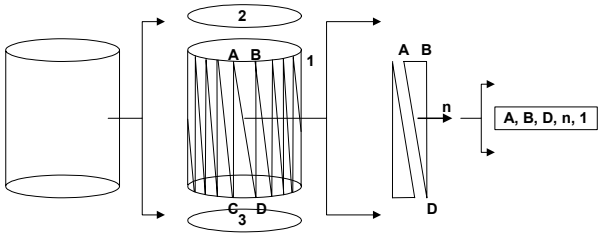

Figure 5 Faceted representation of a cylinder

A complex 3D model is converted to a facet data before it can be rendered. Each facet minimally holds information about the three vertices, the outward normal of the facet and the ID of the face where it came from. Figure 4 and Figure 5 depict the faceted representation of a cube and a cylinder respectively.

5. CLIENT-DISPATCHER-SERVER-RECEIVER

CollabFramework is a flexible framework that modularizes the development of a thin client collaborative system (Mahendra Babu, A. H., et al., 2003). The construction of modules is simplified by providing programming abstractions that are common across modules. Multiple views and dataflow between modules is also supported. The Client-Dispatcher-Server-Receiver pattern shown in Figure 6 is used to achieve the synchronization of the clients. The communication between the clients occurs in the form of distributed command objects. The dispatcher module on the client end (Figure 6) is responsible for forwarding the changes made via the controller to the server's collaboration module. The interface for the dispatcher module is similar to the model. However, the only purpose of the dispatcher is to create appropriate command objects and dispatch them to the server.

As the server receives the command object sent from one of the clients, it makes the same changes in the master model. The server then sends this or a new command object either to the rest of the clients or to all the clients. Thus, all the clients and the server can be synchronized. If the operation corresponding to the command object is to generate a new solid, then the server sends a new command object containing the approximate faceted model to all of the clients. Otherwise, the same command object is sent to the rest of the clients.

On the client end, the receiver module (Figure 6) receives the command object and executes it. This results in the appropriate method on the client's local model being executed. Thus, the models on the other clients are also synchronized.

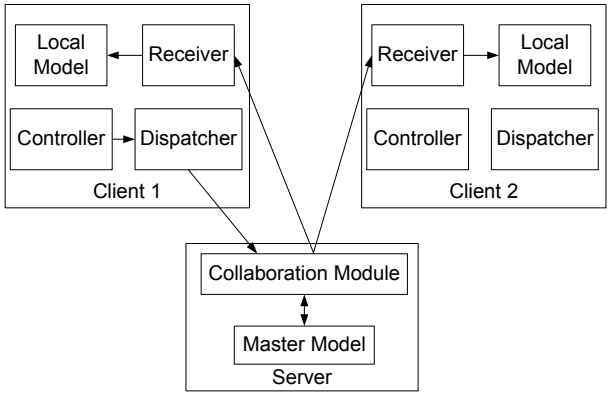

Figure 6 Client-Dispatcher- Server-Receiver pattern

6. SESSION MANAGEMENT

Sessions preserve the state of a discussion on topics selected by session members. Joining a session allows a user to access the data and to collaborate if other users have joined the session. Each session has a channel which is used for transporting the command objects among the participants in the session. The server keeps track of sessions in progress. The clients have the option of joining an already existing session or creating a new session. The clients specify the session to join as an input parameter while connecting. If the session does not exist, a channel for that session is created and returned to the clients by the server. On the other hand, if the session does exist the server returns the user the channel corresponding to that session. While connecting, the clients are sent a copy of the master model on the server. This enables the clients to connect to a session at anytime and still be ensured that they would be in synchronization with the other participants in the session.

7. SERVER SIDE

The server provides support for most of the core computational operations, such as solid modeling. It is the hub where data from all the clients are routed through after processing. The server has the master model that is used to update the approximate version of the shape model on the clients. Since the computationally intensive components reside on the server, the clients are thin, freeing the clients from installing and maintaining software at their end. The server has the following modules (Figure 7).

Collaboration Module – This module manages collaboration, multiple session creation, transfer of editing control and maintaining a master copy of the Model information.

Solid Modeling Module – ACIS™ is used as the solid modeling kernel. ACIS™ provides the software routines to represent three-dimensional solids and to perform operations, such as faceting and interference detection on these solids.

Visualization Module – This module creates facet representation from the boundary representation of the geometry. A polygonal approximation of the BREP model is used for visualization on the client side. The facet representation is generated by approximating the faces of the BREP model by polygons, while maintaining edge consistency between adjacent faces. A face is faceted by subdividing the face in parameter space with a grid whose increments are determined through a user-defined parameter. This parameter determines the accuracy of the faceted representation.

Assembly Module – This module computes the assembly transformations based on the assembly constraints specified between the subcomponents. Also detects the interference problems that could arise during assembly process.

Streaming Module - In CAS, the client requests the data it needs from the server. With DPS, however, as the client's perception changes, the data needed for the client portion of the application changes and requires a more complex request protocol to communicate these changes. The streaming module responds by sending the requested segmented file to all clients in the session simultaneously.

Persistence Module – This module gives persistence to assembly created using the system by storing the data in the database. It uses Java Database Connectivity (JDBC) data access APIs to store the java representation of the product being designed. JDBC™ technology allows access to virtually any tabular data source from the Java programming language.

Database Module – This module is the layer between the server and the database that provides an interface to save and retrieve assembly data from the database.

JNI Module – This module is the bridge between Java and C++ components. ACIS™ is written is C++ and CAS is implemented in Java. Therefore, ACIS™ is connected to the CAS server using the Java Native Interface (JNI).

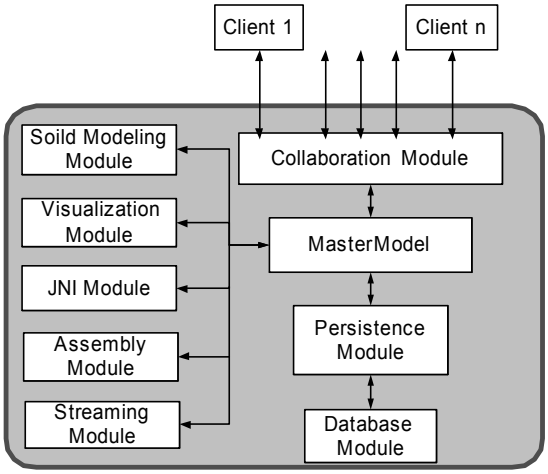

Figure 7 Server side modules

8. COMMUNICATION SECURITY

Security of the data being transmitted during the collaboration session is one of the most important concerns that the users have. The following are the security related issues that can occur in a collaboration system for assembly creation,

- If communications between the clients and server occurs in the clear, an adversary can get access to the data packets in the network and gain knowledge of the CAD files and operations being communicated, which is not desirable.
- The client who has made an assembly modification can refuse having made the modification resulting in controversies.

To incorporate security features, CAS conducts the communications on a secure socket using the Secure Sockets Layer (SSL) or the IETF Transport Layer Security (TLS) protocols, Dierks, C., et al (1999). This eliminates security concerns since the actual CAD file with its embedded intelligence is not transferred to the client machine in the clear. CAS provides the following communication security related services,

- Confidentiality service is provided by encryption and decryption of marshaled data. The Triple Data Encryption Standard (3DES), Data Encryption Standard (1999), cipher algorithm in the Cipher Block Chaining (CBC) mode is used for encryption and decryption. So an adversary will not be able to decipher data packet he obtains from the network.
- Authentication service for the transmitted data is provided using Hash Message Authentication Code (HMAC, 2002). Using authentication the receiver can verify that the data was indeed sent by the proposed sender.
- Integrity service for data communication is guaranteed using hash functions. Secure Hash Algorithm (SHA), Secure Hash Standard (1995) is used for creating the message digest that provides integrity of the transmission. Integrity service detects tampering of data packets by an adversary in the network.
- Non-repudiation service for assembly modifications is provided using digital signatures. When a client makes an assembly modification, along with the corresponding command object, the digital signature of the client is sent to the server. The server logs this command object along with the digital signature after verifying the signature. So if a dispute arises on a later date it can be verified by the server using the logged information. Digital Signature Algorithm (DSA), Digital Signature Standard (2000), using the SHA hash algorithm is used for 128 bit digital signature creation.

9. ASSEMBLY MODULE

CAS currently supports the following types of assembly constraints,

Mate – Two planar surfaces or datum become coplanar and face in opposite directions (Figure 8).

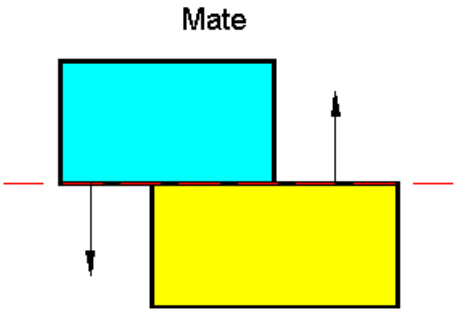

Figure 8 Mate constraint

Align – This can be applied to planar surfaces, revolved surfaces and axes. Planar surfaces become coplanar and face in the same direction (Figure 9). When Align is used on revolved surfaces or axes, they become coaxial (Figure 10).

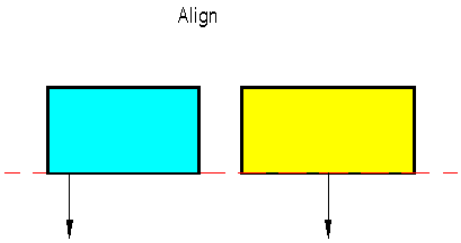

Figure 9 Planar align constraint

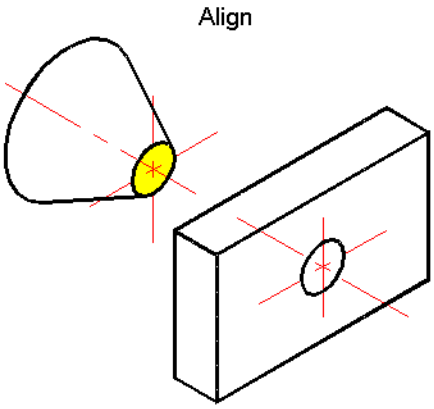

Figure 10 Revolved align constraint

9.1. Assembly Process

The users upload their CAD files into their respective vaults on the server. During the assembly session the users import their subcomponents into the assembly environment. The faceted representation is streamed to all the clients in the session. The master client with the assembly control picks the fixed part and the free part. Then he proceeds to specify the assembly constraints. In the example shown in Figure 11 the user picks the two faces to be mated. The transformation matrix of the free part with respect to the fixed part is computed based on the constraint specified. The matrix contains the translation and rotation transformation needed to orient the coordinate system of the free part with that of the fixed part. The computed transformation is applied to the free part. The free part could be further constrained by specifying another set of constraints (Figure 12). Once all the degrees of freedom of the free part are removed the users can proceed to import the other subcomponents and assemble them. If the assembly constraints result in interference between the subcomponents, the session participants are immediately warned. Figure 13 shows the result of an assembly session.

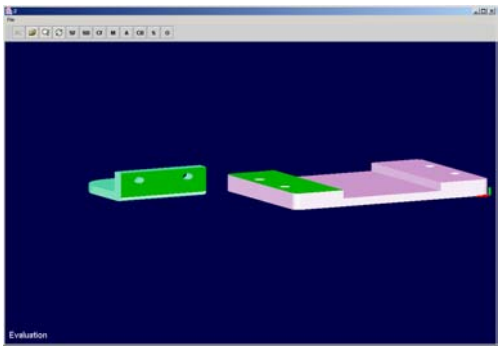

Figure 11 First pair of constraints

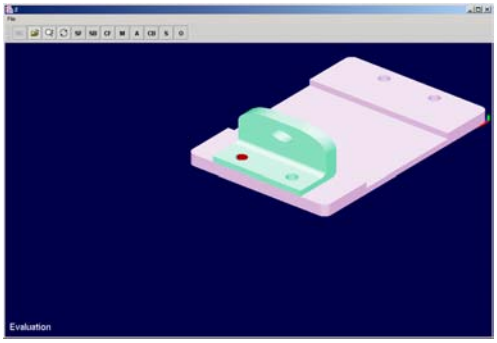

Figure 12 Subcomponents totally constrained

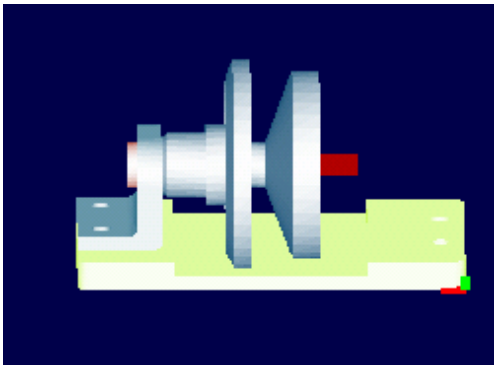

Figure 13 Final assembly

9.2. Persistence to the Assembly

The assembly file created by the user is stored in our proprietary format (.CAS) in the database along with the other project related details. The .CAS file has only the reference to the location in the database of

the subcomponents forming the assembly and their respective transformations. This guarantees the security of the original CAD files uploaded by the users.

10. RESULTS

The faceted representation of the b-rep data is much smaller in size then the CAD design file (around 10% depending on the original document type). PS cycles through the parts in an assembly and requests data in "chunks" (e.g. 10%) until it reaches full resolution. The client would see the assembly initially at a coarse resolution, continuously refining. Since the faceted representation is only about 10% of the original file size and the display data is streamed in 10% chunks, data of only about 1% of the size of the original file needs to be transmitted to the client for the client to have an initial, full view of the complete assembly.

For example, a 100MB assembly requires only 1MB of data be transferred to the client before the entire assembly is displayed. Furthermore, the data is streamed at the part level, so if the assembly is made up of 100 distinct parts, then the first part will display after only 10K of data has been received. Thus regardless of whether users access their 3D CAD models/assemblies via a dial-up or high-speed connection, they can begin working with them immediately. Further upon user interaction DPS takes over and only transfers enough data to accurately visually represent the assembly. Thus CAS is efficient on the client's memory and bandwidth usage.

Figure 14 shows the assembly of the motor and a gear at 50% level of detail. At 50% Level of Detail (LOD) the assembly contains 33498 facets and the facet data structure occupies 129843 bytes. The same assembly upon progressive refinement is shown in Figure 15. At 100% LOD the assembly contains 80649 facets and the facet data structure occupies 336417 bytes.

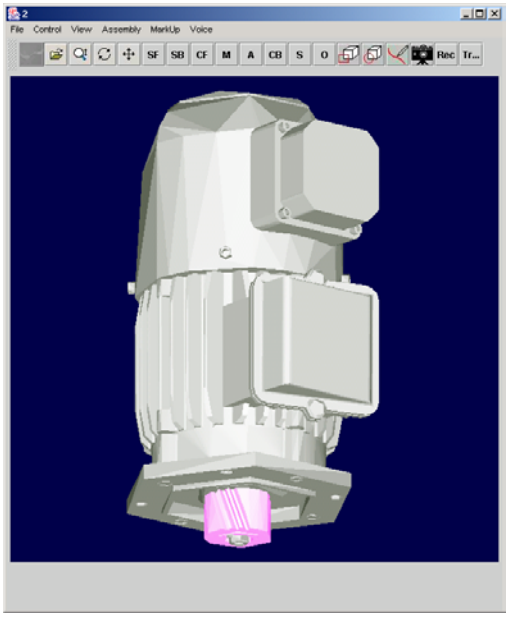

Figure 14 50% LOD

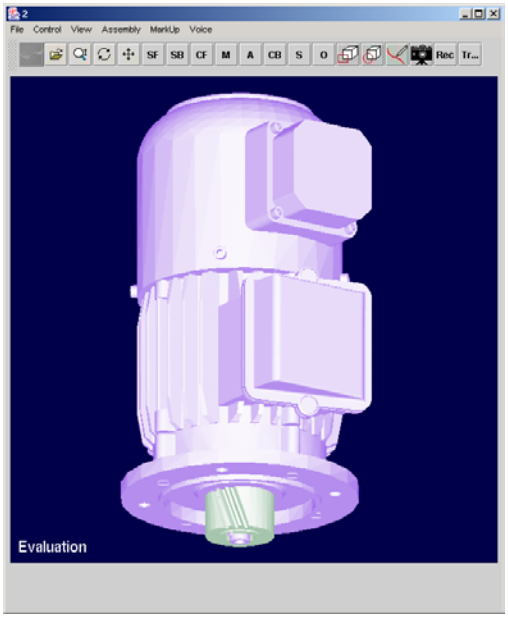

Figure 15 100% LOD

11. CONCLUSIONS

Studies have shown that a fast and accurate access to in-progress designs reduces the product development lead-time and manufacturing cost to a large extent. CAS meets those requirements by providing a secure collaborative assembly system that improves user interaction by streaming the subcomponents. Thus CAS has immense potential to improve the

productivity. To our knowledge CAS is the first assembly system that considers security of the subcomponents and their streaming. As a part of future work the plan is to make the system fault tolerant. Currently work on increasing the set of assembly constraints is also being carried out.

ACKNOWLEDGMENTS

We acknowledge the 21st Century Technology Funds from the state of Indiana and the University Faculty Scholar Award from Purdue University to Professor Karthik Ramani for seeding this research. We acknowledge the National Science Foundation Partnership for Innovation Award (NSF EHR Award# 0227828) for continued support.

REFERENCES

Cimmetry Systems, 2001, www.cimmetry.com.

Cutkosky, M. R., Tanenbaum, J. M., Glicksman, J., 1996, "Madefast: collaborative engineering over the Internet", Communications of the ACM; 39(9):78-87.

Data Encryption Standard (DES), 1999, "No. 46-3 in FIPS, National Institute for Standards and Technology (NIST)", http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

Dierks, T., Allen, C., 1999, "The TLS Protocol Version 1.0. No. RFC2246", http://www.faqs.org/rfcs/rfc2246.html.

Digital Signature Standard (DSS), 2000, "No. 186-2 in FIPS, National Institute for Standards and Technology (NIST)", http://csrc.nist.gov/publications/fips/fips1862/fips186-2.pdf.

Floriani, L. D., Maulik, A., Nagy, G., 1991, "Representation of solid objects by a modularly boundary model", Computer-Aided Mechanical Assembly Planning: 41-80.

Mahendra Babu, A. H., Ramani, K., Joglekar, N., Ganiji, A., (2003) "Flexible Software Framework for Collaboration Systems, Collaborative Design Tools", International CIRP Design Seminar, Grenoble, France, May 12-14, Page 27.

Maxfield, J., Fernando, T., Dew P., 1995, "Distributed virtual environment for concurrent engineering", IEEE Annual Virtual Reality International Symposium, Triangle Park, NC, USA, pp. 162-70.

Park, M., Cutkosky, M. R., 1999, "Framework for modeling dependencies in collaborative engineering processes", Research in Engineering Design; 11:84-102.

RealityWave, 2001, www.realitywave.com.

Regli, W. C., 1997, "Internet-enabled computer aided design", Internet Computing IEEE: 39-50 Jan-Feb.

Requicha, A. G., Whalen, T. W., 1991, "Representations for assemblies", Computer-aided mechanical assembly planning, Boston: Kluwer Academic Publishers.

Roy, U., Bharadwaj, B., Kodkani, S. S., Cargain, M., 1997, "Product development in a collaborative design environment", Concurrent Engineering Research and Applications; 5(4):347-65.

Secure Hash Standard (SHA1), 1995, "No. 180-1 in FIPS, National Institute for Standards and Technology (NIST)", http://www.itl.nist.gov/fipspubs/fip180-1.htm.

Spatial, 2003, www.spatial.com.

Shyamsundar, N., Gadh, R., 2001, "Internet - based collaborative product design with assembly features and virtual design spaces", Computer Aided Design; 33:637-51.

Shyamsundar, N., Gadh, R., 2002, "Collaborative virtual prototyping of product assemblies over the Internet", Computer Aided Design; 34:755-68.

The Keyed-Hash Message Authentication Code (HMAC), 2002, "No. 198 in FIPS, National Institute for Standards and Technology (NIST)", http://csrc.nist.gov/publications/fips/index.html.

Ullman, D. G., 1997, "The mechanical design process", New York: McGraw-Hill.

Ziv, J., Lempel, A., 1977, "A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.