Network-centric CAD

Jae Yeol Lee, Hyun Kim, Joo-Haeng Lee, Nam Chul Do, and Hyung Sun Kim

Concurrent Engineering Research Team
Electronics and Telecommunications Research Institute (ETRI)
jaelee@etri.re.kr

ABSTRACT

Internet technology opens up another domain for building future CAD/CAM environment. The environment will be global, network-centric, and spatially distributed. In this paper, we present a new approach to network-centric virtual prototyping (NetVP) in a distributed design environment. The presented approach combines the current virtual assembly modeling and analysis technique with distributed computing and communication technology for supporting virtual prototyping activities over the network. This paper focuses on interoperability, shape representation, and geometric processing for distributed virtual prototyping. STEP standard and CORBA-based interfaces allow the bi-directional communication between the CAD model and virtual prototyping model, which makes it possible to solve the problems of interoperability, heterogeneity of platforms, and data sharing. STEP AP203 and AP214 are utilized as a means of transferring and sharing product models. In addition, Attributed Abstracted B-rep (AAB) is introduced as 3D shape abstraction for transparent and efficient transmission of 3D models and for the maintenance of naming consistency between CAD models and virtual prototyping models over the network.

Keywords: Virtual prototyping, Assembly modeling, Network-centric CAD, STEP, Feature-based modeling

1. INTRODUCTION

Internet technology opens up another domain for building future CAD/CAD environment. The environment will be global, network-centric, and spatially distributed, which enables product designers to more effectively communicate, obtain, and exchange a wide range of design resources during product development[15]. This improved communication technology has lessened the impact of physical distances on design tasks and has resulted in reconsideration of design activities where design tasks are geographically dispersed. One of these activities is virtual prototyping (VP) that analyzes a product without actually making a physical prototype of the product. The term *virtual* refers to the fact that the design is not yet created in its final form but that only a geometric representation of the object is presented to the user for observation, analysis and manipulation. This prototype does not necessarily have all the features of the final product but has enough of the key features to allow testing of the product design against the product requirements[4].

Virtual prototyping typically involves analyzing computer aided design models for different end applications such as manufacturability or assemblability analysis. Such analyses may be performed via the aid of system modules (or agents) which could be residing in a distributed fashion on the Internet. This distributed environment represents a unique application for WWW. In recent years, the Web has been used very extensively for a large variety of applications. In principle, the Web reduces the distance between: (1) several designers, (2) between designers and software programs, and (3) between different software programs that need information from each other[1,7,15].

Several research efforts have addressed ways in which a computer-network oriented design environment will be able to support product designers and suggest what a computer-based design tool or system should look like in such an environment[3,4,6,8,14,17]. However, these works are conceptual in nature and do not provide well-structured representation and detailed algorithms. For instance, they have not addressed how to distribute necessary processings among distributed components, and how to transmit the shape abstraction of 3D geometric models over the network. The shape abstraction should support efficient transmission of 3D models and contain necessary information for supporting various distributed activities such as assemblability analysis, naming consistency between the server and clients, and 3D geometric constraint solving. For this reason, it is crucial to develop a well-integrated, network-centric, and agential architecture for interoperability and shape abstraction in collaborative and distributed virtual prototyping activities.

In this paper, we present a new approach to network-centric virtual prototyping (NetVP) in a distributed design environment. The presented approach combines the current virtual assembly modeling and analysis technique with distributed computing and communication technology for supporting virtual prototyping activities over the network. The presented approach is implemented in a client/server architecture in which Web-enabled NetVP clients, neutral NetVP server, and other applications communicate with one another via a standard communication protocol for accessing remote objects.

This paper focuses on interoperability, shape representation and geometric processing of 3D CAD models in network-centric virtual prototyping. As a service provider, the NetVP server offers functionalities needed for virtual assembly modeling and analysis, and it is shared among multiple clients. STEP standard and CORBA-based interfaces allow bi-directional communication between the NetVP server and Web-enabled virtual prototyping clients, which makes it possible to solve the problems of interoperability, heterogeneity of platforms, and data sharing. STEP AP203 and AP214 are utilized as a means of transferring and sharing product models. A CORBA-based standard communication protocol is used to link the server and diverse clients in a transparent and modeler-independent fashion. Attributed Abstracted B-rep (AAB) is introduced as 3D shape abstraction of 3D models for transparent transmission and for the maintenance of naming consistency between the NetVP server and clients over the network. AAB is an abstracted and simplified B-rep such that it realizes much data reduction for inexpensive and distributed processings. Moreover, this paper discusses client-side processings needed for distributed virtual prototyping activities such as collision detection and interactive assembly modeling. This means that the NetVP server gives the NetVP client some of the responsibility for processing data. Thus, it is possible to balance necessary processings between the NetVP server and client for minimizing their interactions.

The remainder of this paper is organized as follows. Section 2 overviews the proposed approach. Section 3 presents interoperability, shape representation and geometric processing for network-centric virtual prototyping. Section 4 shows some implementation results. In section 5, we conclude with some remarks.

2. SYSTEM OVERVIEW

The architecture of NetVP is shown in Figure 1. It consists of NetVP Server, DCM Server, Web-enabled NetVP Clients, and CORBA-based standard communication protocol.

The NetVP Server offers the functionality needed for assembly analysis and modeling and it is shared among multiple Web-enabled NetVP clients. It consists of *NetVP Agent Manager*, *NetVP Model*, *NetVP Analyzer*, *STEP Adapter*, *Neutral Feature Model*, and *Solid Modeling Kernel*. The NetVP Agent Manager is a meta object that manages all the connected agents and serves them. An agent takes responsibility for answering requested services to the connected client. The NetVP Model represents a virtual assembly model that contains a set of an assembly hierarchy, assembly parts, relative placements among parts, and other geometric and non-geometric information. The NetVP Analyzer provides various geometric functionalities needed for analyzing the NetVP Model. The STEP Adapter is used to import/export STEP AP203 and AP214 data to/from the NetVP Model. The Neutral Feature Model represents the feature-based part representation of the NetVP Model. It is built upon the Solid Modeling Kernel with a generic naming scheme. The generic naming scheme *generically* names geometric entities over the network. ACIS^{TM} is used as the Solid Modeling Kernel which provides geometric computing functionalities.

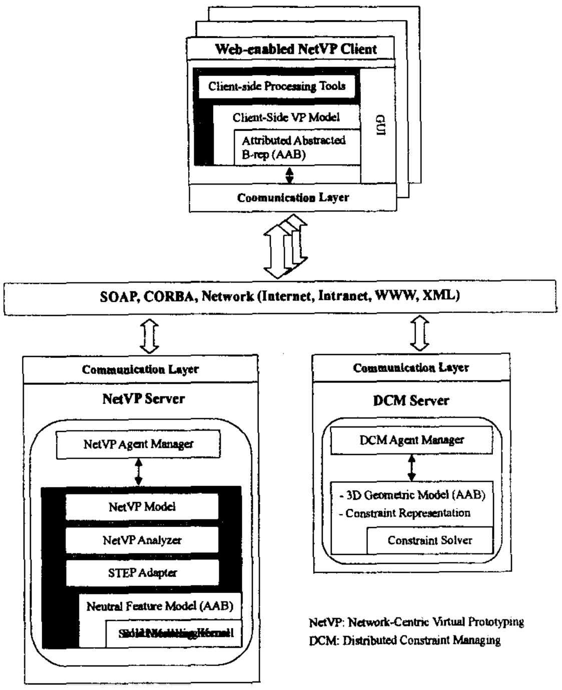

Web-enabled NetVP Client
Client-side Processing Tools
Client-Side VP Model
Attributed Abstracted B-rep (AAB)
GUI
Coomunication Layer
SOAP, CORBA, Network (Internet, Intranet, WWW, XML)
Communication Layer
NetVP Server
NetVP Agent Manager
NetVP Model
NetVP Analyzer
STEP Adapter
Neutral Feature Model (AAB)
Solid Modeling Kernel
Communication Layer
DCM Server
DCM Agent Manager
- 3D Geometric Model (AAB)
- Constraint Representation
Constraint Solver
NetVP: Network-Centric Virtual Prototyping
DCM: Distributed Constraint Managing

Figure 1. System architecture of NetVP

The Distributed Constraint Managing (DCM) Server provides a set of distributed geometric constraint solving mechanisms such as 2D & 3D geometric constraint solving[9,12]. Thus, it can be used in feature-based parametric modeling, dynamic simulation, and assembly modeling. Two major characteristics of the DCM Server are the provision of a broad range of constraints types encountered in design and the ability to solve constraint problems in a distributed manner. Various applications such as constraint-based modeling, assembly modeling, and assembly analysis can run on network-based clients. Each client can view and extract the necessary data required in its application context from the NetVP Model or its shape abstraction. For de-coupled and agential capabilities, it should provide an unambiguous and exact representation of an assembly model in its domain, real-time display of represented parts, fast navigation, user-friendly interface, and various client-side processing tools. For this reason, a 3D shape abstraction called Attributed Abstracted B-rep (AAB) is introduced as a simplified B-rep of the NetVP Model. It consists of approximated faceted data with generic name identifiers such that it is appropriate for inexpensive, transparent and distributed geometric processings: 1) maintaining naming consistency between geometric entities of the server and clients and 2) supporting efficient transmission of the NetVP Model to VP modules.

The CORBA-based standard communication protocol offers the standard view of the modeling server to the layers and applications built on top of it. It forms an interface built along the lines of CAM-I AIS philosophy[2]. It consists of CORBA IDL interfaces that offer a set of assembly and feature modeling services. The member methods of the interface call functions of the NetVP server. These methods provide services to instance primitives, perform Boolean operations and interference checking, sweep planar profiles, delete solids, and perform topological and geometric queries.

3. NETWORK-ENABLED VIRTUAL PROTOTYPING

3.1 NetVP models for interoperability

One of the key issues addressed by the NetVP system is the need for interoperability between NetVP modules and other legacy CAD systems. One of those efforts is STEP standards. In this research, STEP AP203 and AP214 are used as a means of transferring and sharing of assembly product data. In the NetVP server, the STEP Adapter is used for importing/exporting STEP product models to/from NetVP Models.
Currently, the data required by the NetVP are the following:

3.1.1 Assembly structure data

The most important information required is the hierarchy of components of the assembly. This information is supplied in terms of the following:
- Name of the assembly
- Number of children components (for only one level of the tree)
- Names of the children
- Transformation matrix specifying the finally assembled location and orientation of each child component with respect to the parent assembly

The above information could be easily obtained from STEP AP203: Configuration-Controlled Design through the Bill of Materials Unit of Functionality (UoF)[1,7]. This provides the hierarchy of the assembly and the transformations of each part for placement within the assembly.
In addition to the assembly tree structure, a graph-based structure is also used in the NetVP

client. Each node contains the component information, and each arc stores complete information about the type of attachments between the parts and information about the mating surface of the part.

3.1.2 Assembly constraints

Constraint satisfaction is crucial to maintain design integrity and to simulate physical constraints. STEP AP203 is limited to only representing assemblies as a collection of 3D objects positioned and oriented in space. There is no provision within AP203 to capture logical relationships between parts, the mating feature relationships, and part functionality, although work is ongoing in the area of constraints for STEP to represent constraints between geometric entities within a model as well as constraints between components.

In the NetVP system, the DCM Server plays a core role in defining and solving geometric and non-geometric constraints in a distributed environment. Thus, other VP modules can communicate with the DCM Server such that various problems related to constraints can be handled effectively.
The constraint information is supplied to the NetVP system in terms of the following:

- Constraint type: coordinate system, against, fits, mate, distance, angle, etc.
- Geometry type: point, line, circle, plane, cylinder, sphere, etc. (these geometries are defined in the 3D space. For example, a circle can be defined by a set of a center point, an axis normal, and radius.)
- Engineering constraint: a set of equations

For example, assembly constraints can be assigned within the NetVP client by selecting faces on the components of the NetVP client model. Messages including geometries and constraints are sent to an agent of the DCM server to establish connections between the referenced 3D geometries in the solver and the corresponding component faces in the remote NetVP client model. The interoperability between the two models is done via generic name identifiers in the AAB.

When the user requests the DCM agent to solve the constraint system, the agent analyzes the constraint system and solves it. Finally, it returns the evaluated solution to the NetVP client. In the case of assembly constraint solving, the evaluated solution is used to reposition the components of the assembly with respect to the base component. Note that the client can also request for the repositioning of the components of the NetVP server model, if necessary. Thus, the NetVP can support 2-way communication between the server and client.

3.1.3 Feature-based component geometry

The component or part geometry resides in both models: NetVP server model and client-side model. The neutral feature-based part representation built on top of B-rep is used as the component geometry in the NetVP Server model, and the AAB-based part representation is used in the client-side model. The neutral feature model is a solid model with generic naming identifiers. On the other hand, the AAB-based part representation is the shape abstraction of the feature-based part representation.

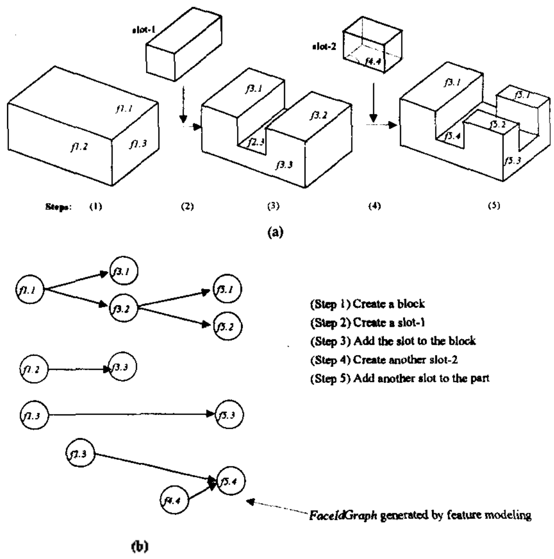
Generic name identifiers are assigned by a generic naming scheme. The scheme generically names geometric entities that are invariant over geometric processings such as topological changes and Boolean operations. It associates each model with a *FaceIdGraph* which is updated every time the topology of the model changes as shown in Figure 2[10,11,13]. The scheme keeps information about how faces of a model were created, split, merged, trimmed and deleted. The FaceIdGraph is a directed acyclic graph consisting of *FaceIdNodes*. The incoming edges to a FaceIdNode represent information about the ancestors of this face, and the outgoing edges represent what happened to the face.

FaceIds are the principal types of face identifiers. Regularized solid models are 3D volumes bounded by a set of faces. Edges and vertices are considered to be intersections of bounding faces, and thus, *EdgeIds* and *VertexIds* are defined in terms of their adjacent FaceIds.

Each face in the boundary of a solid model is assigned a unique FaceId. The FaceId of a face *f* is defined by three components:

FaceId(f) = [*stepId or componentId*, *faceIndex*, *surfaceType*]

where *stepId* is the ID of a modeling step during which *f* was created (or componentId is the Id of a component in the assembly), *faceIndex* is the face index of the face *f* within the step *stepId*, and *surfaceType* is the type of the underlying surface of the face. For detailed information, see the references 11 and 13.


slot-1
slot-2
f1.1
f1.2
f1.3
f2.3
f3.1
f3.2
f3.3
f4.4
f5.1
f5.2
f5.3
f5.4
Steps: (1) (2) (3) (4) (5)
(a)
(Step 1) Create a block
(Step 2) Create a slot-1
(Step 3) Add the slot to the block
(Step 4) Create another slot-2
(Step 5) Add another slot to the part
FaceIdGraph generated by feature modeling
(b)

Note: In this example we use notation *fm.n* denoting a face created during *StepId m* and with index *n* at the step *m*.

Figure 2. FaceIdGraph generated by incremental feature modeling operations

The neutral feature-based part representation with the generic naming scheme plays the following several important roles in the NetVP;

1. It provides naming consistency for interoperability between geometric entities of the NetVP server and their clients over the network.
2. It is general enough to be used in the feature-based part model and assembly component model. In the case of importing STEP AP203 data, the generation of the neutral feature model for each assembly component is simple. Each component is considered a feature-based part consisting of only a base feature (called a *rigid* part). Thus, all the stepIds of topological entities are the same to the componentId in the assembly. However, in the NetVP, we also support feature-based part modeling called NetFEATURE[13] where an assembly component can be designed by feature-based modeling operations. In this case, the assembly can be collaboratively designed in the integrated and distributed environment. Some of the components can be designed in dispersed locations. Whenever changes the component in the NetFEATURE, the updated specification is sent to the referenced component within the NetVP client. The assembly model is then re-evaluated according to the updated specification. Thus, the generic naming scheme is necessary for both modeling and interoperability of the assembly.
3. It identifies topological entities of solid models in such a way that the same entities can still be identified after the models have been re-evaluated from feature-based modeling operations in the NetFEATURE.

This shows how the NetVP server, clients, and DCM server communicate one another for distributed virtual prototyping activities.

3.2 AAB for shape abstraction

The choice for shape abstraction in the NetVP can range from a polyhedral model in which a shape is represented explicitly as a collection of 2D triangular facets to a high-level boundary representation scheme such as those used by a typical solid modeler[4]. The advantage of a polyhedral representation is that it is possible to store in parallel the model in a triangulated form. This eliminates the need for a separate triangulation step since the triangular faceted representation is readily available to the collision detection and rendering routines which require data in this form. A significant disadvantage of the polyhedral representation is that this representation loses all the topological and geometric information inherent to the B-rep such that various geometric processings cannot be supported. Another approach for shape representation is to use a traditional boundary representation as a geometric engine. However, there is no solid modeler written in Java, and it is too heavy to be used in a Web-enabled NetVP client.

The Attributed Abstracted B-rep is introduced as shape abstraction of 3D geometric models. The AAB has two types of representations: 1) Face-based AAB and 2) Edge-based AAB. The face-based AAB consists of FacetedFaces and the Edge-based AAB consists of FacetedEdges [13]. The face-based AAB is a B-rep consisting of triangulated polyhedrons that approximate the surface representation of the part, while the edge-based AAB is a B-rep of faceted edges that approximate the wireframe representation. Each FacetedFace consists of a FaceId for the generic name, a FaceEq for the face equation, a VertexList for all points and their normals, and an IndexedTriangleList for all indexed triangles as shown in Figure 3. Note that each FaceId represents a generic name of a face and it is used as a communication identifier between the server and client. As a result, AAB can provide real-time displays, navigation, and

various geometric interrogations such as mass property, area, and ray test in the client.
However, the AAB does not support all the necessary functionalities that can be provided by the original geometric modeling kernel since it does not have topological information and geometric query functions. In those cases, those functionalities can be accessed via the standard communication protocol.

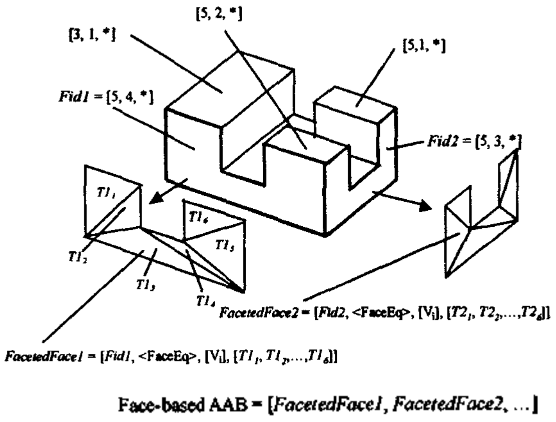

[5, 2, *]
[3, 1, *]
[5,1, *]
Fid1 = [5, 4, *]
Fid2 = [5, 3, *]
T1₁
T1₂
T1₃
T1₄
T1₅
T1₆
FacetedFace2 = [*Fid2*, <FaceEq>, [Vi], [*T2₁*, *T2₂*,...,*T2₆*]]
FacetedFace1 = [*Fid1*, <FaceEq>, [Vi], [*T1₁*, *T1₂*,...,*T1₆*]]
Face-based AAB = [*FacetedFace1*, *FacetedFace2*, ...]

Figure 3. Face-based AAB of the model in Figure 2

3.3 Client-side processing

Client-side processing is an important issue on the Web or network-centric environment. This means that the server gives the client some of the responsibility for processing data. That is, the client must provide necessary functionalities for domain dependent client-side processings. However, all these functionalities cannot be provided by the client itself[13]. Thus, it is important to balance necessary processings between the server and client for minimizing their interactions.

3.3.1 Processing tools

The NetVP client provides various client-side processing tools such as design, interaction, and analysis tools as shown in Figure 4. The design tool provides feature-based part modeling operations that are necessary to create the components of an assembly. In addition, it provides a tool for constraint-based assembly modeling. The interaction tool provides navigation, selection and manipulation of the assembly model. The NetVP client also provides several analysis tools such as interactive assembly modeling, rapid collision detection, measuring & clearance checking, cross sectioning, transforming, and markup tools.

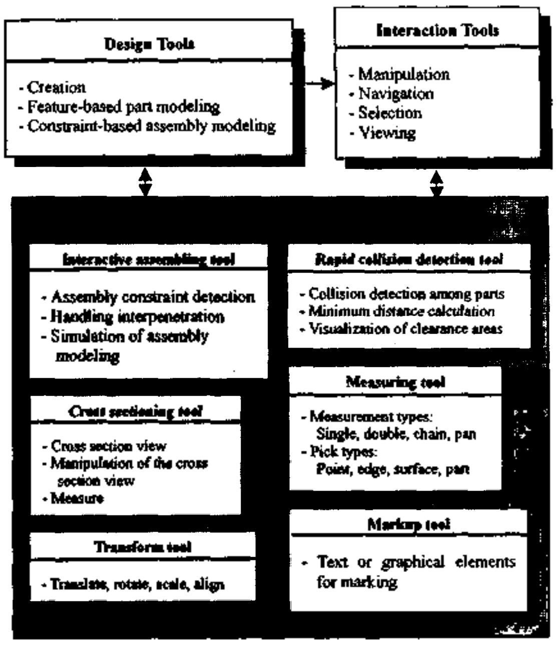

Design Tools
- Creation
- Feature-based part modeling
- Constraint-based assembly modeling
Interaction Tools
- Manipulation
- Navigation
- Selection
- Viewing
Interactive assembling tool
- Assembly constraint detection
- Handling interpenetration
- Simulation of assembly modeling
Rapid collision detection tool
- Collision detection among parts
- Minimum distance calculation
- Visualization of clearance areas
Measuring tool
- Measurement types: Single, double, chain, pan
- Pick types: Point, edge, surface, part
Cross sectioning tool
- Cross section view
- Manipulation of the cross section view
- Measure
Markup tool
- Text or graphical elements for marking
Transform tool
- Translate, rotate, scale, align

Figure 4. Client-side processing tools

The measuring tool measures the distances between entities or sets of parts in the assembly model. The clearance checking tool identifies the clearance of parts. Clearance is the minimum distance between two parts. Parts that are in contact have a clearance distance of 0. When a pair of parts has a clearance distance of 0, the parts are either in contact with each other or one of the parts penetrates the other. We can determine the nature of the contact, display the precise regions of the contact and display a cross section view of the contact using the cross sectioning tool. The transforming tool is to change a model by translating, rotating, or scaling. We can transform a part or group of parts in the assembly to change its position, orientation, and size, independently of the rest of the assembly. The transforming and constraint solving tools can change the geometric configuration of the assembly. Thus, if needed, the client can communicate with the server to modify the configuration of the NetVP server model.

However, some operations cannot be done in the client. Then, the client can request the NetVP server to provide the necessary processings. For that reason, the NetVP server provides basic interfaces for geometric processings: 1) creation/deletion, 2) topological queries, 3) properties and 4) modifications[1,6,17]. Each interface of the communication protocol has a one-to-one map with the underlying solid modeling API function. Also depending on the functionality of the modeler, some functions may have to be added or deleted if the entire functionality of that modeler needs to be used. Assembly analysis can be done by the combination of the basic functions of the NetVP server.

Among the geometric processing tools mentioned above, the rapid collision detection and interactive assembly modeling tools are fundamental components for virtual prototyping activities in the client-side.

4. SYSTEM IMPLEMENTATION

The NetVP system has the architecture of multiple clients, DCM server, Session Manager, and NetVP server as shown in Figure 5. The servers have been implemented in C++, and the client has been implemented in Java. The communication between the server and clients is done via the CORBA-based communication interface. For effective visualization of the client model, the scene graph is used to provide an interface to the Java3D graphics library. The scene graph constructs a part description specific of the AAB to the Java3D library.

4.1 NetVP examples

A typical virtual prototyping procedure takes the following steps. First, the user uses a World-Wide Web browser to enter the NetVP site and downloads a Java applet (1). The client in the applet connects to an agent in the NetVP server (2). Then, the agent connects to a database server for transaction management (3). Finally, the user is ready for network-centric virtual prototyping.
Thus, the user can load or save existing assembly models from/to the database server. For loading data from the database server, the user is asked if he/she wants to check these data out of the database server. When the checkout is validated, the data are transmitted to both the modeling server and the client. Figure 5 shows what's the result of the proposed approach, and what is currently being implemented.

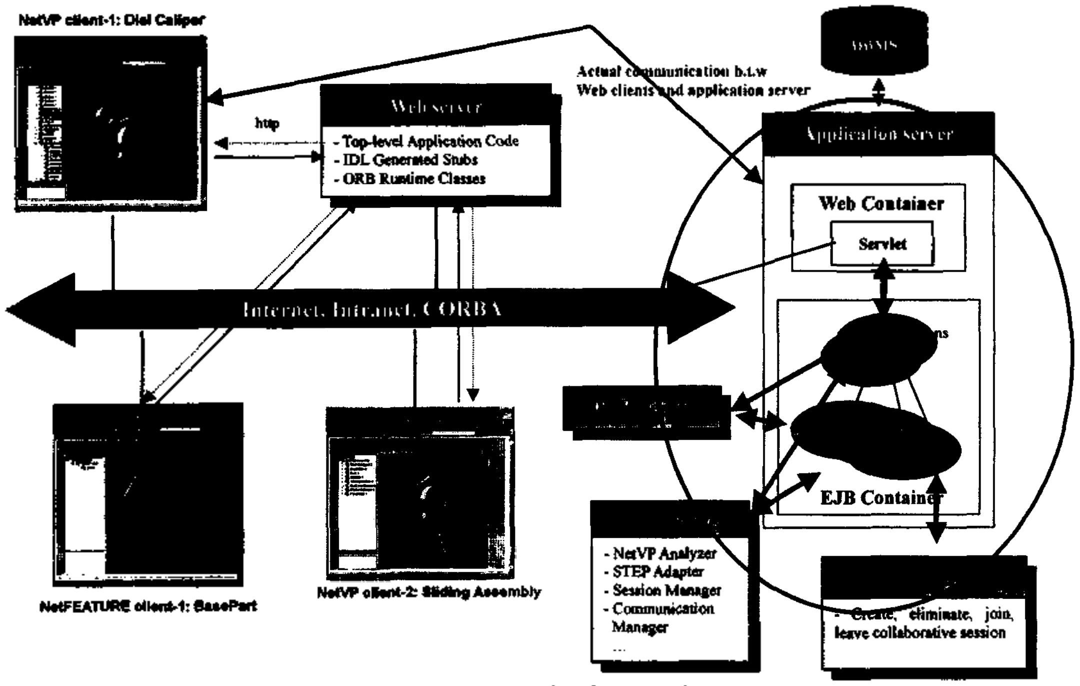

NetVP client-1: Dial Caliper
DBMS
Actual communication b.t.w
Web clients and application server
Web server
http
- Top-level Application Code
- IDL Generated Stubs
- ORB Runtime Classes
Application server
Web Container
Servlet
Internet, Intranet, CORBA
EJB Container
- NetVP Analyzer
- STEP Adapter
- Session Manager
- Communication Manager
...
- Create, eliminate, join, leave collaborative session
NetFEATURE client-1: BasePart
NetVP client-2: Sliding Assembly

Figure 5. System implementation

Figure 6 and 7 show various kind of geometric processings done in the client-side. Figure 6 shows the visualization and analysis of STEP AP203 data representing a drill assembly. The drill has been assembled in a commercial CAD software, SolidWorks™. Then, the assembly has been saved as a STEP physical file. Finally, the STEP file has been imported into the NetVP system. In the NetVP system, each part is converted to a neutral feature model by attaching generic identifiers. Then, the AAB of the neutral model is transmitted to the NetVP client. Moreover, the assembly hierarchy is also extracted from the STEP data and transmitted to the NetVP client. Figure 6 also shows a good example of virtual prototyping in a client-side: cross sectioning. Figure 7 shows an example of assembly constraint solving and simulation: a universal joint assembly. Since we cannot obtain constraint information from the STEP standard, currently, the user can assign assembly constraints in the Web-enabled client. Then, DCM server solves the constraint system and sends the evaluated solution to the client. By changing values of constraints, we can simulate the assembly as shown in Figure 7.

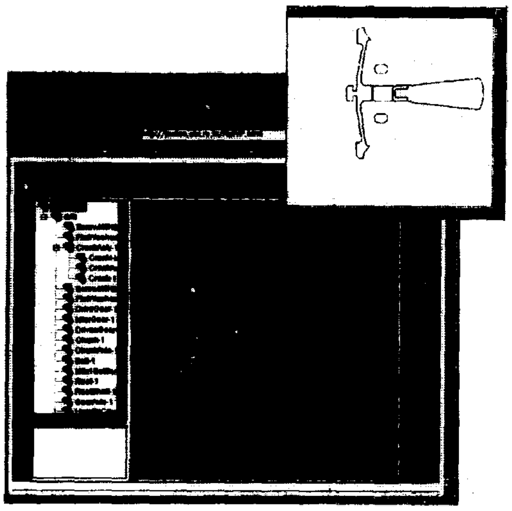
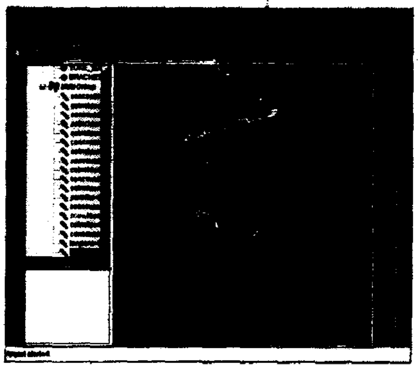


Figure 6. STEP visualization and analysis (cross sectioning)


(a)

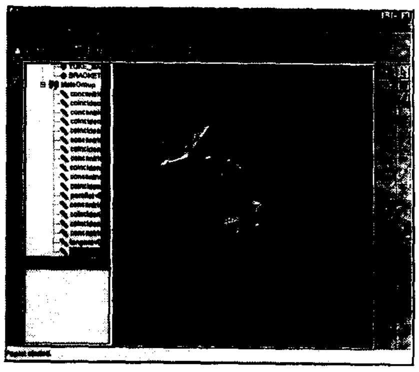

(b)

Figure 7. Assembly modeling and simulation

4.2 Compressed AAB

As explained before, the AAB maps the *exterior* surfaces of a 3D product model and creates a streamlined envelope part that retains accurate information about the surface and mass properties of the original model. By producing lightweight representations of large assemblies, the AAB allows engineering teams to exchange design data with *minimal* load both on system resources and on the network. For example, Table 1 shows how the AAB can abstract and reduce the original assembly model: comparing the data size between the original model and AAB model.

Table 1. AAB metrics of the model shown in Figure 6

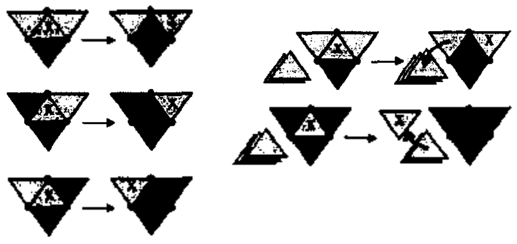
Export Type	Amount of model captured	Quality setting	File size (MB)	Size saving
Original assembly	STEP	▨	13.8	
AAB (1083 faces)	9394 triangles	1	0.91	93.3%
	13628 triangles	4	1.21	91.2%
	26406 triangles	7	2.01	85.4%

Note that we found that there might be about 90% reduction in data size. Moreover, we can refine the quality of the AAB for several purposes since the NetVP client can communicate with the NetVP server and update the AAB (e.g., each part can have a different quality setting). The quality can be refined by the combination of surface tolerance, normal tolerance, grid aspect ratio, and maximum facet edge length. For example, for collision detection, the parts to be analyzed can have higher level of detail, but the others have lower level of detail.

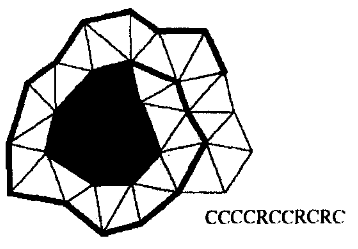
Further, the AAB makes it possible to share only surface and mass property information for such tasks as visualization and assembly studies without disclosing proprietary design information. This shows what's the advantage of using the AAB in network-based CAD applications.

For supporting more efficient interoperability between the server and client, we need to develop geometry compression and decompression algorithms for AAB. As mentioned before, visualization information takes the most part of AAB. Thus, it is possible to compress/decompress the connectivity of arbitrary triangle meshes of AAB and its vertex information. For that reason, we utilize the Edgebreaker scheme for triangle compression [16]. The Edgebreaker compression is guaranteed to encode any triangulated planar graph of triangles with at most 1.84*t* bits. It stores the graph as a CLERS string-a sequence of *t* symbols from the set {C, L, E, R, S}, each represented by a 1, 2 or 3 bit code as shown in Figure 8. For more information, readers are referred to see Reference 16.

Figure 9 shows the compression ratio between AAB and compressed AAB. Although the size of the compressed AAB can vary according to vertex quantization bits, it just takes less than 10 % of the size of AAB. Usage of compressed AAB and AAB allows to exchange design data with minimal load on both system resources and on the network.

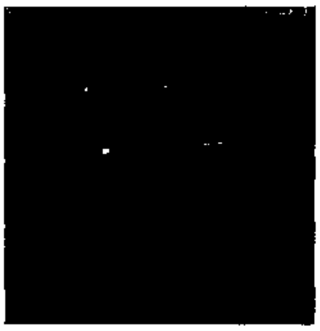


(a) Edgebreaker's compression state machine



(b) A typical starting phase of Edgebreaker's compression

Figure 8. Edgebreaker's compression scheme



Quantization bits	AAB	Compressed AAB	Size saving
16 bits	49.3KB	5.28KB	89.3%
14 bits		4.87KB	90.1%
12 bits		4.26KB	91.4%
10 bits		3.52KB	92.9%
8 bits		2.81KB	94.3%

Figure 9. Compressed AAB metrics

5. CONCLUSION

This paper presents a new approach to virtual prototyping in a distributed design environment. This paper focuses on the interoperability, shape representation, and geometric processings of 3D CAD models for network-centric virtual prototyping. It also describes how the system architecture should be for cost-effective, flexible, and portable distributed modeling. Some advantages of this research are summarized as below:

1) The feature-based NetVP server acts as a service provider for assembly modeling and analysis in a distributed environment. Moreover, it provides a scheme for consistent naming of topological entities such that it can maintain the persistent relationship between geometric entities of the server and clients.
2) The Attributed Abstracted B-rep (AAB) is introduced as shape abstraction for an interoperable common geometric model and for client-side processing. AAB is an abstracted and simplified B-rep such that it is appropriate for inexpensive and distributed processings. AAB has been successfully applied to 3D assembly modeling and analysis and to geometric constraint solving.

However, there are also some issues to be integrated further in the future research as below:

1) We are currently integrating

NetFEATURE[13] with NetVP. The integration will be done based on the central database server. The server will play a major role in distributing and communicating among NetFEATURE and NetVP clients.

2) It requires much work for collaborative modeling environment. It will require the integration of additional support tools with the system such as email, video conferencing, XML, SOAP, etc.

REFERENCES

1. Angster, S., Lyons, K., Hart, P., and Jayaram, S., "Interoperability of assembly analysis applications through the use of the open assembly design environment". *Proc. ASME Design Engineering Technical Conference*, Atlanta, Georgia, EIM-5680, 1998.
2. CAM-I, Application Interface Specification AIS 2.0. Technical Report R-90-PM-03, Consortium for Advanced Manufacturing, 1990.
3. Fernando, T., Wimalaratne, P., and Tan, K., "Constraint-based virtual environment for supporting assembly and maintainability tasks", *Proc. ASME Design Engineering Technical Conference*, Las Vegas, Nevada, CIE-9043, 1999.
4. Gadh, R. and Sonthi, R., "Geometric shape abstractions for internet-based virtual prototyping". *Computer-Aided Design*, Vol. 30, No. 6, pp. 473-386, 1998.
5. Gottschalk, S., Lin, M., and Manocha, D., "Obb-tree: A hierarchical structure for rapid interference detection". *Proc. ACM Siggraph*, pp. 171-180, 1996.
6. Han, J.H. and Requicha, A.A.G., "Modeler-independent feature recognition in a distributed environment". *Computer-Aided Design*, Vol. 30, No. 6, pp. 453-463, 1998.
7. Hardwick, M., Spooner, D.L., Rando, T., and Morris, K.C., "Sharing manufacturing information in virtual enterprises". *Communications of the ACM*, Vol. 39, No. 2, pp. 46-54, 1996.
8. Jayaram, S., Connacher, H.I., and Lyons, K.W., "Virtual assembly using virtual reality techniques". *Computer-Aided Design*, Vol. 29, No. 8, pp. 575-584, 1997.
9. Kim, J., Kim, K., Choi, K., and Lee, J.Y., "Solving 3D Geometric Constraints for Assembly Modeling". *International Journal of Advanced Manufacturing Technology*, Vol. 16., No. 11, pp. 843-849, 2000.
10. Kripac, J., "A mechanism for persistently naming topological entities in history-based parametric solid models". *Computer-Aided Design*, Vol. 29, No. 2, pp. 113-122, 1997.
11. Lee, J.Y., Kim, H., and Han, S.-B., "Web-enabled feature-based modeling in a distributed design environment". *Proc. ASME Design Engineering Technical Conference*, Las Vegas, Nevada, DFM-8941, 1999.
12. Lee, J.Y. and Kim, K., "A 2D geometric constraint solver using DOF-based graph reduction". *Computer-Aided Design*, Vol. 30, No. 11, pp. 883-996, 1998.
13. Lee, J.Y., Kim, H., and Kim, K., "A Web-enabled approach to feature-based modeling in a distributed and collaborative design environment". *Concurrent Engineering: Research and Applications*, Vol. 9, No. 1, pp. 74-86, 2001.
14. Martino, T.D., Falcidieno, B., and Hasinger, S., "Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment". *Computer-Aided Design*, Vol. 30, No. 6, pp. 437-452, 1998.
15. Regli, W.C., "Internet-enabled computer-aided design". *IEEE Internet Computing*, pp. 39-50, 1997.
16. Rossignac, J., "Edgebreaker: connectivity compression for triangle meshes". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 1, pp. 47-61, 1999.
17. Shah, J.J., Dedhia, H., Pherwani, V., and Solkhan, S., "Dynamic interfacing of applications to geometric modeling services via modeler neutral protocol". *Computer-Aided Design*, Vol. 29, No. 12, pp. 811-824, 1997.