

Role-Based Viewing
for
Secure Collaborative Modeling

Christopher D. Cera
Taseong Kim
Ilya Braude
JungHyun Han
and
William C. Regli

Technical Report DU-CS-03-01
Department of Computer Science
Drexel University
Philadelphia, PA 19104
February 2003

Role-Based Viewing for Secure Collaborative Modeling

Christopher D. Cera¹ Taeseong Kim² Ilya Braude¹ JungHyun Han² William C. Regli^{1*}

¹ Geometric and Intelligent Computing Laboratory
Dept. of Computer Science, College of Engineering
Drexel University
Philadelphia, PA 19104
<http://gic1.mcs.drexel.edu>

² Computer Graphics Laboratory
School of Information and Communications Engineering
Sung Kyun Kwan University
Suwon, 440-746, Korea
<http://graphics.skku.ac.kr>

Abstract

Information security and assurance are new frontiers for collaborative design. In this context, information assurance (IA) refers to methodologies to protect engineering information by ensuring its availability, confidentiality, integrity, non-repudiation, authentication, access control, etc. In collaborative design, IA techniques are needed to protect intellectual property, establish security privileges and create “need to know” protections on critical features. Aside from 3D watermarking, research on how to provide IA to distributed collaborative engineering teams is largely non-existent.

This paper provides a framework for information assurance within collaborative design, based on a technique we call role-based viewing. Such role-based viewing is achieved through integration of multi-resolution geometry and security models. 3D models are geometrically partitioned, and the partitioning is used to create multi-resolution mesh hierarchies. Extracting a model suitable for access rights for individual actors within a collaborative design environment is driven by an elaborate access control mechanism.

1 Introduction

Information assurance (IA) refers to methodologies to protect and defend information and information systems by ensuring their availability, confidentiality, integrity, non-repudiation, authentication, access control, etc. [24] In collaborative design, IA is mission-critical. Suppose a team of designers working collaboratively on a 3D assembly model. Each designer has a different set of security privileges and no one on the team may have the “need to know” the details of the entire design. In collaboration, designers must interface with others’ components, but do so in a way that provides each designer with only the level of information he or she is permitted to have about each of the components. For example, one may need to know the exact shape of some portion of the component (including mating features) being created by another designer, but not the specifics of any other aspects of the component. Such a need can also be found when manufacturers out-source designing a sub-system: manufacturers may want to hide critical information of the entire system from suppliers.

The authors believe that IA represents a new problem that needs to be addressed in the development of collaborative CAD systems. There are several significant scenarios we envision.

Protection of sensitive design information: As noted above, designers may have “need to know” rights based on legal, intellectual property, or national security requirements.

Collaborative supply chains: Engineering enterprises out-source considerable amount of design and manufacturing activities. In many situations, an organization needs to provide vital design data to one partner while protecting the intellectual property of another partner.

*URL: <http://www.cs.drexel.edu/~wregli>; Email: regli@drexel.edu

Multi-disciplinary design: Designers of different disciplines working on common design models often suffer from cognitive distraction when they must interact with unnecessary design details that they do not understand and cannot change. For example, an aircraft wheel well [2] is a complex and confusing place in which electronics, mechanical, and hydraulics engineers all interact in close quarters with vast amounts of detailed design data. These interactions could be made more efficient if the design space could be simplified to show each engineer just the details he or she needs to see.

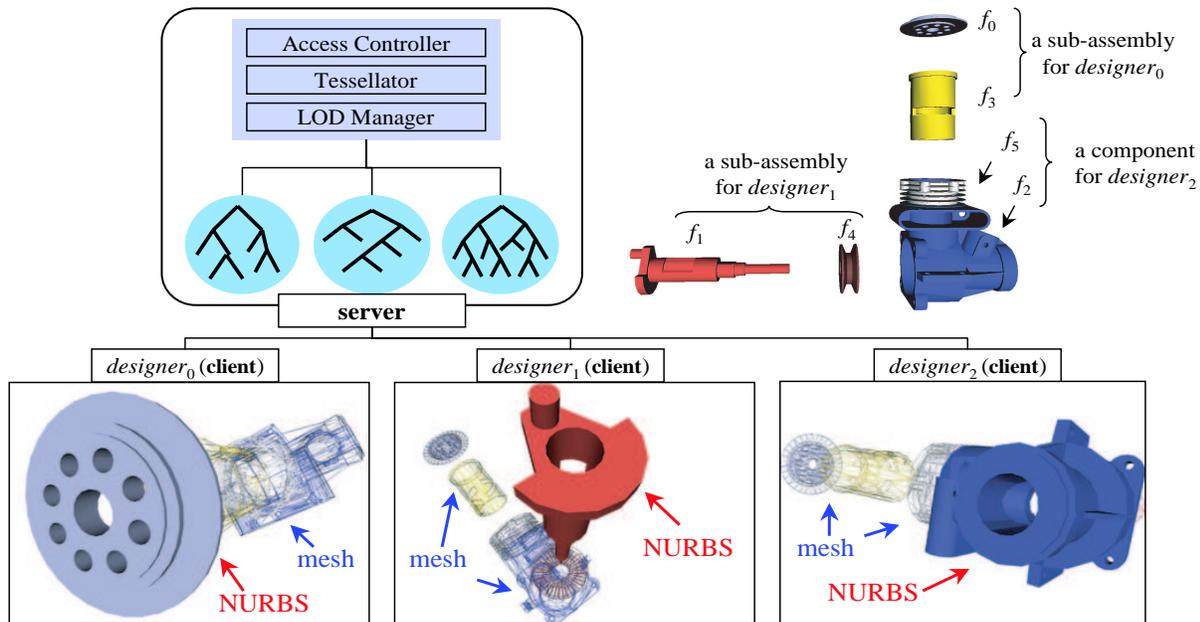


Figure 1: Secure Collaborative Design System Architecture

This paper develops a new technique for *role-based viewing* in a collaborative 3D assembly design environment, where multiple users work simultaneously over the network. It is an incorporation of IA into collaborative CAD. Among various issues in IA, *access control* is critical for the purpose. We will present a combination of *multi-resolution geometry and access control models*. Specifically, we introduce:

Security framework for collaborative CAD: The access control framework presented in this paper provides a specification for actors(users), roles, and their authorized permissions on objects.

Artifact access control: The designed objects, or solid models, are partitioned into a set of regions. Each of these regions, whether a point, a patch, a component, or a sub-assembly, is related with a set of roles. The access control model is not limited to geometric regions, and is general enough to be used for feature and constraint data.

View generation: Given an actor and access authorization database, a 3D model is generated for viewing which does not compromise sensitive information about model geometry, topology or behavior.

Figure 1 illustrates the conceptual architecture of the secure collaborative design system, which can be stated as follows:

- An assembly model consists of a set of component parts, possibly grouped into sub-assemblies.
- Each component part is represented as a NURBS-based boundary model.
- Design is performed collaboratively by a group of designers working on different (possibly geographically distant) workstations. A standard client-server architecture is assumed, where the *collaborative CAD server* maintains and synchronizes the master design model. Individual designers work on different sets of components locally, at their *collaborative CAD clients*.

- The collaborative CAD server tessellates the master model into polygon meshes. Multi-resolution mesh hierarchies are constructed, which are used in generating *role-based views* for designers working at the client hosts.
- Depending on the accessibility privilege of a designer, an appropriately simplified model is extracted from the multi-resolution hierarchies, and provided to the designer at the collaborative CAD client. In Figure 1, the client side NURBS model being designed is shaded whereas the other components (in meshes) of the master model are rendered as wire frames, just for illustration.
- When a component part or sub-assembly gets modified, the server reconstructs only the corresponding (changed) portion of the hierarchy, and then passes these updates to the other clients according to their accessibility privileges.

Following sections will discuss the key issues in developing such a secure collaborative design system. Aside from digital 3D watermarking, research on how to provide IA to distributed collaborative designers is largely non-existent. The authors believe that this work represents the first attempt to provide IA to computer-aided design and collaborative engineering.

2 Related Work

2.1 Collaborative Design

There has been a vast body of work on concurrent engineering and collaborative design. In our view, this research can be loosely grouped into two categories, which we will call *data centric* and *interaction centric*.

Data centric research focuses on collaborative data sharing or knowledge sharing. Historically, research of this kind emerged simultaneously from engineering, artificial intelligence and database communities. In contrast, interaction centric approaches deal with real-time or synchronous collaboration among people in the design process. These environments would usually require 3D graphical interfaces. In other cases, the environment consist of computer-supported cooperative work (CSCW) tools coupled with design systems.

The subset of existing work most relevant to our efforts is interaction centric, dealing with real-time 3D collaboration and communication. Distributed Virtual Environments (DVEs) [1, 5, 7, 11, 15] have been developed for real-time interactions between distributed collaborators in a number of different domains. Immersive environments such as CAVE [4] have been developed which also support real-time interaction, but they do not necessarily support collaborative CAD. Conner et al. [3] directly addressed the use of distributed VR for collaborative design, but in this work the design data was largely static and not worked on synchronously by multiple users.

2.2 Information Assurance and Access Control in 3D Models

Current research on information assurance incorporates a broad range of areas such as data availability, confidentiality, integrity, non-repudiation, authentication, access control, etc. In CAD domain, information assurance research has been partially addressed through the development of 3D digital watermarking [16, 17, 19]. It has been used to ensure the *integrity* of a model as well as provide a foundation for proof of copyright infringement.

This paper focuses on *access control*. Access broadly refers to a particular mode of operation such as read or write. Access control is the process of limiting access to resources of a system only to authorized users, programs, or processes, and therefore preventing activity that might lead to a breach of the system's security.

Access control assumes that *authentication* of users has been verified. Authentication services are used to correctly determine the identity of a user. If the authentication mechanism of a system has been compromised, then the access control mechanism which follows will certainly be compromised.

In CAD and collaborative design contexts, few research results on access control have been reported. A most relevant work in the domain of collaborative assembly design can be found in Shyamsundar and Gadh [23]. A component (or a sub-assembly) is partitioned into *interface features* and an *envelope* which approximates the component. Such an envelope may be a convex hull, a bounding box/sphere, or a special bounding volume that comprises of the external faces of the component. Their work could be taken as a simple implementation of information-hiding techniques, but lacks an elaborate access control mechanism. Further, it will be desirable to provide finer-grained levels of detail than envelopes.

2.3 Multi-resolution Techniques

Polygon meshes lend themselves to fast rendering algorithms, which are hardware-accelerated in most platforms. Many applications, including CAD, require highly detailed models to maintain a convincing level of realism. However, the number of polygons is often greater than that we can afford. Therefore, *mesh simplification* is adopted for efficient rendering, transmission, and various computations. The most common use of mesh simplification is to generate *multi-resolution* models or various *levels of detail* (LOD). For example, near objects are rendered with a higher LOD, and distant objects with a lower LOD. Thanks to LOD management, many applications such as CAD visualization can accelerate rendering and increase interactivity. A most recent survey on mesh simplification can be found in [14].

The most popular polygon-reduction technique is *edge collapse* or simply *ecol* (more generally, vertex merging or vertex pair contraction) where two end vertices are collapsed into a single one. Repeated applications of *ecol* generate a simplified mesh.

Vertex split or simply *vsplit* is the inverse operation of *ecol*. Hoppe proposed *progressive mesh*(PM) [9], which consists of a coarse base mesh (created by a sequence of *ecol* operations) and a sequence of *vsplit* operations. Applying a subset of *vsplit* operations to the base mesh creates an intermediate simplification. The *vsplit* and *ecol* operations are known to be fast enough to apply at runtime, therefore supporting dynamic simplification.

Previous works on mesh simplification and LOD techniques often mention the possibility of applying the techniques to collaborative design. To date, however, their use has been limited to the areas such as redundant geometry reduction, realtime rendering, streaming 3D data over the network, etc.

3 Role-based Viewing

In *role-based viewing*, each user sees a shared 3D model in a distinct resolution, which is determined by the user's *role*. The following subsections discuss the tools required for role-based viewing.

3.1 Access Control Policies

Existing *access control policies* are briefly surveyed in this subsection. Access control policies commonly found in contemporary systems can be classified as follows [22].

- Discretionary Access Control
- Mandatory Access Control
- Role-based Access Control

Discretionary Access Control (DAC) was originally introduced by Lampson [12], where the access of a user to an object is governed on the basis of authorizations that specify the access mode (e.g. read, write, or execute) the user is allowed on the object. Typically, the owner of an object has discretion over what users are authorized to access the object. DAC policies do not impose any restriction on the usage of information once a user has acquired it, and therefore have the drawback that they do not provide real assurance on information flow.

Mandatory Access Control (MAC) [18] policies control dissemination of information by associating users and objects with *security levels*. The security level associated with an object reflects the *sensitivity* of the information, i.e. the potential damage that could result from unauthorized disclosure of the information. The security level associated with a user reflects the user's *trustworthiness* not to disclose sensitive information to users not cleared to see it. MAC policies assert that a user can access an object only if the user has a security level higher than or equal to that of the object. For example, suppose that the security levels consist of Top Secret(TS), Secret(S), Confidential(C), and Unclassified(U), and that $TS > S > C > U$, where $>$ denotes "has a higher security level than." An S-level user can then access a C-level object, but not a TS-level one. This is often called the "read down" principle. For the other principle called "write up," readers are referred to [22].

In Role-Based Access Control (RBAC) [21], system administrators create *roles* according to the job functions in an organization, grant permissions (access authorizations) to the roles, and then assign users to the roles. The permissions associated with a role tend to change much less frequently than the users who fill the job function that role represents. Users can also be easily reassigned to different roles as needs change. These features have made RBAC attractive, and numerous software products such as Microsoft's Windows NT currently support it.

Our security framework is essentially based on embodiment of a MAC policy within an RBAC framework. It will be implemented as an *access matrix* as discussed in Section 3.3.

3.2 Role-based View

A *role-based view* is a tailored 3D model which is customized for a specific user based on the roles defining the user’s access permissions on the model. In this way, the role-based view does not compromise sensitive model information which the user is not allowed to see (or see in detail).

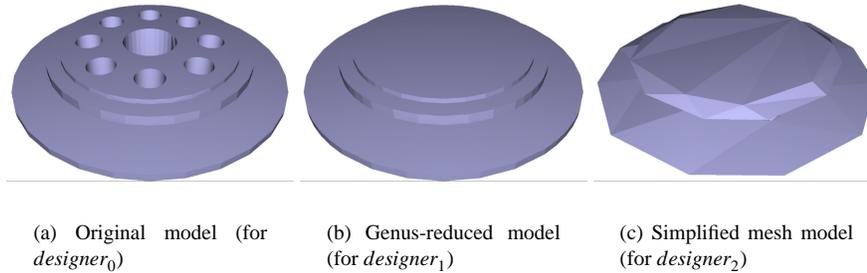


Figure 2: Role-based View Examples of f_0

Consider the component f_0 in Figure 1, which is being edited by $designer_0$. Suppose that $designer_0$ wants to hide the design details of f_0 from other participating designers, i.e. $designer_1$ and $designer_2$. Our solution to the problem is to present f_0 to them in some “lower” resolutions. Figure 2 shows three different resolutions or LODs of f_0 . Figure 2-(a) is a full-resolution model, which $designer_0$ sees and may also be presented to, for example, project supervisors.

The set of holes in f_0 might be critical features which $designer_0$ wants to hide from $designer_1$. Then, all holes are removed from the original model, and the model in Figure 2-(b) is presented to $designer_1$. Suppose that $designer_2$ is a supplier from another organization. Then, the model in Figure 2-(b) can be again simplified to generate the crude model in Figure 2-(c), which just presents the outline of f_0 to $designer_2$. Those are examples of role-based views. Our system provides an appropriate resolution to each designer according to the designer’s roles.

Roles, $R = \{r_0, r_1, \dots, r_m\}$, are abstract objects that define both the specific users allowed to access resources and the extent to which the resources are accessed.

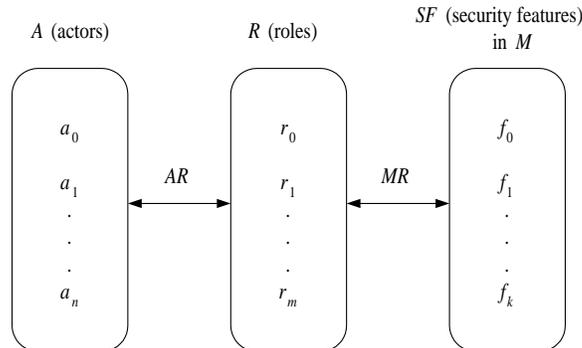


Figure 3: Actors, Roles and Features

The engineers (designers, process engineers, project supervisors, etc.) correspond to a set of *actors* $A = \{a_0, a_1, \dots, a_n\}$,

each of which will be assigned to a set of roles. *Actor-Role Assignment*, AR , is a many-to-many relation of actors to roles: $AR \subseteq A \times R$. See Figure 3.

The entire assembly design is represented as a *solid model* M . A collaborative engineering environment enables multiple engineers (actors) to simultaneously work with M . Let $b(M)$ represent the boundary of M . *Model-Role Assignment*, MR , is a many-to-many relation assigning points on $b(M)$ to roles: $MR \subseteq b(M) \times R$, where each point on $b(M)$ is assigned to at least one role, i.e., $\forall p \in b(M) \exists r \in R, (p, r) \in MR$.

It is impractical to assign $b(M)$ to roles point-by-point. Hence, we define a set of *security features*, $SF = \{f_0, f_1, \dots, f_k\}$, where each f_i is a topologically connected point set on $b(M)$ and $\cup SF = b(M)$. Such *security features* can correspond to assembly features, mating features, or other function-based features of M . The Model-Role Assignment can then be simplified to be the relation associating security features with roles: $MR \subseteq SF \times R$. See Figure 3.

Suppose that, for example, AR assigns actor a_3 to roles r_{20}, r_{23} , and r_{75} . This entitles a_3 to view (and perhaps change) the security features assigned (by MR) to these roles. Portions of $b(M)$ not assigned to these roles, however, are “off limits” to actor a_3 .

Partitioning $b(M)$ into security features SF can be done either by the project supervisor (working as an administrator) or by the designers in charge of the components or sub-assemblies to be partitioned. Boundary partition can be done through sub-assembly by sub-assembly, component by component, form/design feature by feature (in the context of feature-based design), NURBS surface by surface, or even patch by patch. In Figure 1, the assembly model is partitioned into 6 security features f_0, f_1, f_2, f_3, f_4 and f_5 , where $\{f_3, f_4, f_5\}$ is a set of mating features.

3.3 Access Matrix

Access matrix is a popular *conceptual model* that specifies the rights that each user possesses for each object. In a large system, the access matrix is usually enormous in size and sparse. Therefore, compact access control lists (ACL) are often used to implement the access matrix.

	f_0 (TS)	f_1 (C)	f_2 (S)	f_3 (U)	f_4 (U)	f_5 (U)
r_0 (TS)	w	r	r	r	r	r
r_1 (S)	r	w	r	r	r	r
r_2 (C)	r	r	w	r	r	r

Figure 4: Access Matrix

In the collaborative CAD context, however, an access matrix is constructed and maintained “for each design session,” and consequently the matrix is dense because every component/sub-assembly is supposed to be visible to virtually all participating designers (probably in different LODs). Therefore, we adopt a matrix implementation as illustrated in Figure 4, which is for the collaborative assembly design example in Figure 1. There is a row in this matrix for each role, and a column for each security feature. For simplicity, only three roles, r_0, r_1 and r_2 , are created.

Such an access matrix is obviously an RBAC implementation. To embody a MAC policy in it, let us associate both roles and security features with *security levels* using the simple hierarchy of $TS > S > C > U$. (In fact, boundary partitioning is followed by associating each feature with a specific security level.)

Each cell of the access matrix distinguishes between *read* and *write* authorizations. It is reasonable to assume that write permission of a feature is exclusively given to a single role. In contrast, read permissions of a feature should be given to all roles. For the remainder of this paper, we focus on read permissions.

A typical scenario within this RBAC+MAC system would be that, for example, a C-level feature is visible to S-level role whereas a TS-level feature is invisible. Rather than this “all or nothing” read permissions, our objective is to assign a “continuous” *degree of visibility* between a feature and a role, i.e. the method presented in this paper may generate a “full” resolution version of the C-level feature and a “lower” resolution version of the TS-level feature to the S-level role.

The administrator not only constructs the access matrix and registers it into an authorization database, but also performs Actor-Role Assignment AR . Suppose that, in the simple example of Figures 1 and 4, actors $designer_0, designer_1$, and $designer_2$ are assigned to roles r_0, r_1 , and r_2 respectively. Let us focus on f_0 . The write permission given to r_0 implies the full read permission, regardless of the security levels associated to r_0 and f_0 . Therefore,

$designer_0$ who has the write permission on f_0 sees a full resolution of f_0 . This is the view given in Figure 2-(a). In contrast, $designer_1$ takes r_1 's security level S , and it is lower than the level TS of f_0 . Therefore $designer_1$ should see a simplified model. It might be the view given in Figure 2-(b). Finally, $designer_2$'s security level C is far lower than the level TS of f_0 , and therefore $designer_2$ might see a drastically simplified model, which might be the view given in Figure 2-(c). Such a "continuous" role-based viewing technique is discussed in the next section.

4 Role-Based View Generation

To an actor a , role-based viewing presents a new model M' , which is generated from the original assembly model M such that its security features are appropriately obfuscated based on the actor a 's roles. If the roles give the actor full permissions to see certain features, then the resulting model M' includes those features with the same fidelity as in M ; if not, the features must be obfuscated so as to hide from a what a does not have permissions to see.

The input to role-based viewing consists of an actor a , the Actor-Role Assignment (AR), access matrix, and multi-resolution mesh hierarchies for the entire assembly. As AR and the access matrix have been previously discussed, this section focuses on multi-resolution mesh hierarchies, and how to implement RBAC+MAC using the hierarchies.

4.1 Multi-resolution Mesh Hierarchy

Numerous mesh simplification approaches have been proposed in computer graphics literature. Some key features that distinguish among the approaches are as follows.

- topology-preserving vs. topology-modifying: Topology preserving simplification algorithms preserve manifold connectivities at every step, but topology modifying ones do not necessarily do so and therefore permit drastic simplification.
- static/discrete vs. dynamic/continuous: Static simplification usually computes LODs offline during preprocessing and rendering algorithms select an appropriate LOD at runtime. Dynamic simplification creates a data structure encoding a continuous spectrum of detail, and a desired LOD is extracted from this structure at runtime. It also supports progressive transmission.

For rendering, an object's topology is less important than its overall appearance. We also need an algorithm capable of drastic simplification since runtime performance is crucial in our system. Therefore, topology-modifying simplification is a reasonable choice. Further, topology modification such as *genus reduction* often plays an important role in hiding the design-detail of a component/sub-assembly.

In a collaborative design system where a number of designers collaborate simultaneously, it is more storage-efficient to have a single dynamic/continuous hierarchy rather than multiple discrete LODs. Further, an appropriate LOD need often be transmitted to each client depending not only on each designer's access privilege but also on each client's computing capability (triangle or polygon budget!). A continuous hierarchy guarantees extremely fine granularity in the sense that a distinct LOD can be presented to each actor. Therefore, the progressive mesh (PM) discussed in Section 2.3 is a reasonable choice.

4.2 Genus Reduction in Feature-based Design

A problem of PM is that it assumes manifold topology, and consequently is not compatible with topology-modifying simplification. Its solution can be found by utilizing *feature-based design* capabilities, which most of contemporary CAD systems support.

Let us consider solid modeling. Features are classified into positive/additive and negative/subtractive features. The negative features lead to depressions such as holes. In the first stage of our simplification process, such negative features may be removed from the original model, and then topology-preserving simplification (*ecol*) is applied at the second stage. Note that the topology-preserving simplification enables drastic polygon reduction because genus is reduced at the first stage. Such an integration of feature-based genus reduction and topology-preserving simplification is much faster than topology-modifying simplification algorithms such as [6]. Figure 2-(b) shows a model with negative features removed, and Figure 2-(c) shows the result of applying mesh simplification to the model in Figure 2-(b).

4.3 Role-based Viewing integrated with MAC

A role-based view is generated “security features by features.” We distinguish between *genus-reducible* security features from others. In the context of feature-based design, for example, a security feature is *genus-reducible* if it contains a non-empty set of negative design features whose dimensions are below some predetermined threshold values. For a *genus-reducible* security feature, two mesh data structures are constructed: one is a plain mesh for the entire security feature, and the other is a PM of the *genus-reduced* model. If a security feature is not *genus-reducible*, it is just represented as a PM.

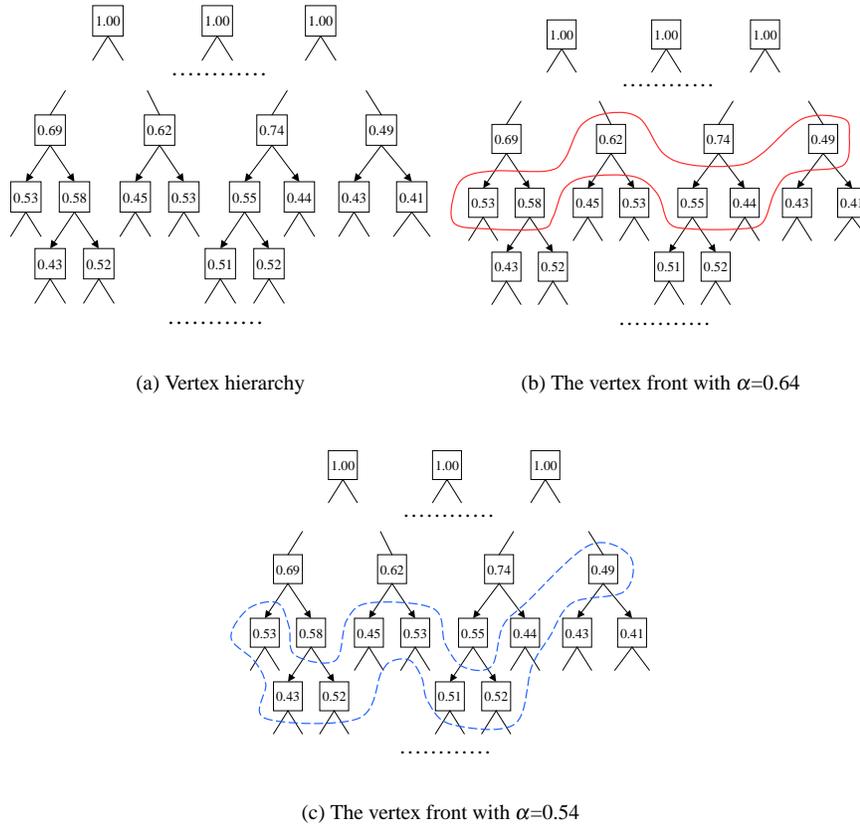


Figure 5: Progressive Mesh Hierarchy

We have a PM per a security feature. As discussed in Section 2.3, a PM data structure consists of a base mesh and a list of *vsplit* nodes. The *vsplit* list can be conceptually illustrated as a forest of binary vertex trees as shown in Figure 5-(a). Each PM node corresponds to a vertex. Therefore, a *vsplit* operation splits a vertex into two new vertices corresponding to its two children.

The problem of how much of a security feature is made visible to a role is reduced to the task of what subtrees of its PM to select, or how to choose a “vertex front” [10] of the PM. (All vertices of a simplified mesh extracted from a PM constitute a vertex front in the PM’s hierarchical structure, as depicted in Figures 5-(b) and -(c).) The solution to the task requires understanding of the mesh simplification method we adopted.

Garland and Heckbert [8] proposed a mesh simplification algorithm based on *quadric error metrics* (QEM). It proceeds by repeatedly merging vertex pairs, each of which is not necessarily connected by an edge, i.e. it modifies topology. We use a slight modification of the algorithm: QEM coupled with *ecol*, not the general vertex merging. A QEM is associated with each vertex and represents the sum of the squared distances from the vertex to the neighboring

triangles. Error caused by an *ecol* operation is easily obtained by summing the QEMs of the two vertices being merged, and the sum is assigned to the new vertex as a QEM. All *ecol* candidates are sorted in a priority queue, and the simplification algorithm selects the edge with the “lowest error” and then performs *ecol*. The algorithm then updates the errors of all edges involving the merged vertices and repeats the simplification.

As *ecols* are selected basically in order of increasing errors, the inverse operations *vsplits* are roughly listed in order of decreasing error values. In PM, all leaf nodes have error 0, and one of root nodes will have the maximum error e_{max} . The range $[0, e_{max}]$ is normalized into the range $[0, 1]$. Such a normalized error is depicted for each node in Figure 5. (For implementation purpose, the error values of all root nodes are made 1.00.)

MAC policy allows us to have as many levels of security as needed. Let us denote the highest level as l_{max} , the lowest level as l_{min} , the level assigned to a role as l_r , and the level assigned to a security feature as l_f . Our MAC policy asserts that, if $l_r \geq l_f$, the full-resolution version of the feature is presented: (1) If the security feature is genus-reducible, the plain mesh for the entire security feature is transmitted. (2) Otherwise, the vertex front is formed with all “leaf nodes” of the security feature’s PM.

When $l_r < l_f$, the vertex front should be composed of “internal nodes” of PM. Let us define the *degree of visibility* α mentioned in Section 3.3. If $l_r < l_f$, α is set using a *distance metric*, which is defined as follows:

- $(l_f - l_r - 1)/(l_{max} - l_{min})$ if feature-based genus reduction has been performed
- $(l_f - l_r)/(l_{max} - l_{min})$ otherwise

The reason for two metrics will be discussed soon. The point is that, as the second metric clearly says, a larger α value is computed when the distance between l_f and l_r is longer. Obviously, the larger α value is, the lower resolution is required. In fact, degree of visibility is a misnomer, and α actually denotes the degree of *invisibility*.

Note that the α value computed as above is also normalized into the range of $[0, 1]$. Therefore, it can be directly used to determine the vertex front in PM where *ecol* errors have also been normalized. In the list implementation of PM, simple list operations are invoked to select a subset of *vsplit* nodes whose error values are greater than or equal to α . The base coarse mesh followed by the selected *vsplit* nodes are transmitted to clients, and a simplified mesh is rendered. Figure 5-(b) shows the vertex front determined by $\alpha=0.64$, and Figure 5-(c) by $\alpha=0.54$. Compare the two vertex fronts. As 0.64 is larger than 0.54, a lower resolution should be presented for the case of $\alpha=0.64$. Therefore the vertex front of $\alpha=0.64$ lies higher than that of $\alpha=0.54$.

There can be many other ways to obtain the vertex front. A simpler way is to make α determine the percentage of *vsplit* nodes. For example, if α is 0.7, 30% (=1-0.7) of the *vsplit* nodes are selected. However, our experiments showed that the elaborate mechanism based on QEM values leads to “more expectable” degradation of the model fidelity.

Let us now discuss why we need two metrics for α . Suppose that $l_f - l_r = 1$, i.e. the role’s security level is just one degree lower than that of the security feature. If feature-based genus reduction has been performed, the PM represents an already-simplified model. Therefore, it is reasonable, when $l_f - l_r = 1$, to present the full PM, i.e. the vertex front should consist of all leaf nodes of PM. It is achieved when $\alpha=0$. For that purpose, we subtract 1 from $l_f - l_r$ to set $\alpha=(l_f - l_r - 1)/(l_{max} - l_{min})$ to 0.

When the level difference between a role and a security feature is extremely large, we could make the security feature completely deleted or replaced with a simple convex hull or bounding box. For example, if $\alpha=1$, i.e. if $l_f=l_{max}$ and $l_r=l_{min}$, we could simply make the security feature invisible. It is implementation dependent.

5 Implementation and Results

To test the approach we have described in this paper, a prototype system was developed. For collaborative design, it is imperative to make the system independent of platforms and operating systems. Previous versions of our system were written in Java3D, but we found that Java3D incurred too much overhead when trying to present multiresolution surfaces in real-time. The revised system has been simultaneously developed using OpenGL on Solaris 2.7-2.8 and Windows, and using both Mesa and nVidia’s native OpenGL drivers on Linux operating systems.

The environment we developed is divided into two stages: authoring and viewing. The authoring stage allows a designer to assign a {label, permission}-tuple to parts, assemblies, or individual facets. These models are stored in a VRML-like format within the design repository. The normalized permissions $[0.0 - 1.0]$ were used to indicate a percentage of vertices in the original model. We envision that a production system would compute all necessary

resolutions during the design stage for approval by an authority, and possibly store the “secure” models explicitly. In situations where this is inadequate, a supervised technique, such as semiautomatic simplification, can be used.

The trivial authentication mechanism we have created allows an administrator to specify users, roles, and hierarchical relations. At the time a designer wants to view a model, they must declare their identity so their role associations can be derived. Based upon the roles associated with a designer and the model features, a *role-based view* is generated. We used a single administrative account to modify permissions in the model repository. There are numerous administrative configurations which have been presented by Sandhu [20]. The goals and constraints of the collaboration will dictate how comprehensive the role administration requirements should be.

We have implemented our own topology-preserving QEM-based simplification algorithm. For the experiments in this paper, we chose to collapse only vertex pairs which are connected by an edge. For *role-based viewing* in CAD, it is difficult to envision a situation where collapsing disconnected vertex pairs would yield a more viable result. The QEM algorithm is passed each part, or connected region of a part with an equivalent {label, permission} set of tuples. Since these regions are disjoint, they can be simplified and transmitted in parallel.

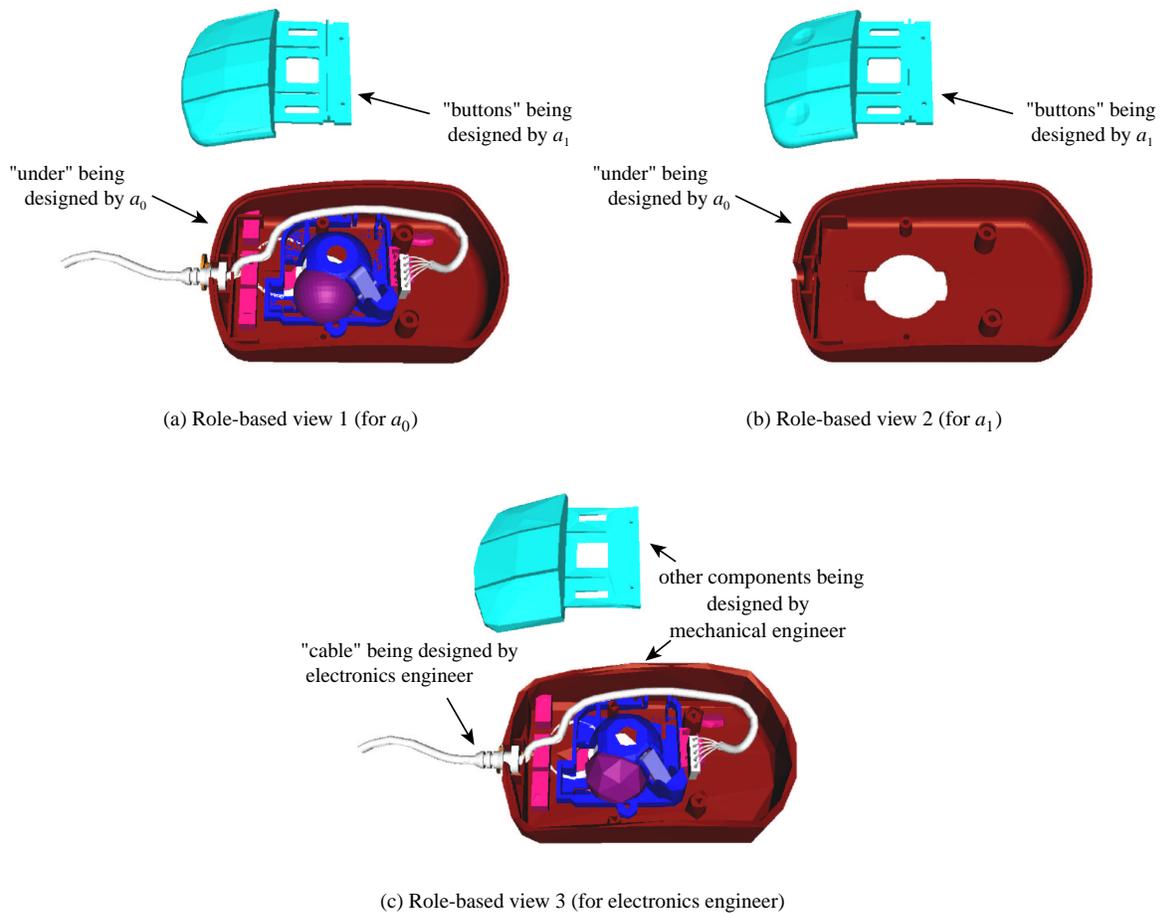


Figure 6: Test Results for a Mouse Assembly

Example: Mouse Assembly Figure 6 shows three role-based views of a mouse. Figures 6-(a) and -(b) assume that actor a_0 is editing the lower part named “under” and actor a_1 is editing the upper part named “buttons.” Figure 6-

(a) is a view for a_0 : “under” is presented in a full resolution.

In this example, both left and right buttons of the mouse are designated as separate security features, and assigned high security levels. (This example uses security levels defined in the real-value range $[0,1]$.) Suppose that the security level of a_0 is much less than those of the left and right buttons. Therefore, both the left and right buttons of the mouse are hidden in Figure 6-(a) presented to a_0 .

In contrast, Figure 6-(b) is a view for a_1 , the “buttons” designer. A number of components in “under” are completely deleted in the figure. It is because a_1 is associated with the minimum security level whereas the hidden components are associated with the maximum level.

In Figure 6-(c), we assume that the mouse is collaboratively designed by an electronics engineer and a mechanical engineer. The electronics engineer is in charge of only the white-colored “cable,” and all other components are designed by the mechanical engineer. This figure is a view for the electronics engineer, and therefore all components except the cable are presented in low resolutions.

6 Conclusions and Future Work

This paper has presented a new technique, *role-based viewing*, for collaborative 3D assembly design. By incorporating security with collaborative design, the costs and risks incurred by multi-organizational collaboration can be reduced. Aside from digital 3D watermarking, research on how to provide security issues to distributed collaborative design is largely non-existent. The authors believe that this work is the first of its kind in the field of collaborative CAD and engineering.

Our security framework is essentially an embodiment of a MAC policy within an RBAC framework, and is implemented as an access matrix. Recent works on RBAC proposed sophisticated structures such as role hierarchy [21]. Hierarchies are a natural means for structuring roles to reflect an organization’s lines of authority and responsibility. Further, roles can *inherit* permissions from other roles. We are currently investigating the possibility of replacing the access matrix by a role hierarchy.

Our present implementation focuses on meshes only. In the same access control framework, however, we can augment meshes with NURBS. Then, multi-resolution analysis (MRA) based on wavelet decomposition would be needed. With this extension, a low resolution NURBS model is transmitted to clients, and it enables *design comments* or *design suggestions* from the other designers: They can suggest some design changes to the owner of the model, for example, by moving some control points of the NURBS model.

We have proposed an automatic simplification technique to degrade the fidelity of a model enough to satisfy the access-control requirements of a collaborative session. In some cases, however a form of semiautomatic simplification [13] might need to be employed. Semiautomatic simplification is a means of supervising the mesh reduction process by editing the order of *ecols*, selecting regions where more or less simplification is necessary, and directly manipulating the vertex hierarchy. One disadvantage of semiautomatic simplification is that parameters of the simplification will need to be stored with the model, since these cannot be automatically derived.

Acknowledgements This work was supported in part by National Science Foundation (NSF) Knowledge and Distributed Intelligence in the Information Age (KDI) Initiative Grant CISE/IIS-9873005; CAREER Award CISE/IIS-9733545 and Office of Naval Research (ONR) Grant N00014-01-1-0618. Additional support has been provided by the Korea Research Foundation Grant KRF-2001-013. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organizations.

References

- [1] John W. Barrus and Richard C. Waters. Locales: Supporting Large Multiuser Virtual Environments. *IEEE Computer Graphics and Applications*, 16(6):50–57, 1996.
- [2] Sean Callahan and Jeff Heisserman. *A Product Representation to Support Process Automation*, In *Product Modeling for Computer Integrated Design and Manufacture*, Edited by M.J. Pratt, R. D. Sriram, and M.J. Wozny. Chapman and Hall, 1996.

- [3] D. Conner, M. Cutts, R. Fish, H. Fuchs, L. Holden, M. Jacobs, B. Loss, L. Markosian, R. Riesenfeld, and G. Turk. An Immersive Tool for Wide-Area Collaborative Design. TeamCAD, the First Graphics Visualization, and Usability (GVU) Workshop on Collaborative Design. Atlanta, Georgia, May 12-13, 1997.
- [4] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. ACM Press, 1993.
- [5] J. M. S. Dias, R. Galli, A. C. Almeida, C. A. C. Bello, and J. M. Rebordao. mWorld: A Multiuser 3D Virtual Environment. *IEEE Computer Graphics and Applications*, 17(2):55–65, 1997.
- [6] J. El-Sana and A. Varshney. Controlled Simplification of Genus for Polygonal Models. In *IEEE Visualization*, pages 403–412, 1997.
- [7] H. Eriksson. MBONE: The Multicast Backbone. *Communications of the ACM*, 37(8):54–60, August 1994.
- [8] Michael Garland and Paul S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.
- [9] Hugues Hoppe. Progressive Meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM Press, 1996.
- [10] Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198. ACM Press/Addison-Wesley Publishing Co., 1997.
- [11] Sankar Jayaram, Uma Jayaram, Yong Wang, H. Tirumali, K. Lyons, and P. Hart. VADE: a Virtual Assembly Design Environment. *IEEE Computer Graphics and Applications*, 19(6), 1999.
- [12] Butler Lampson. Protection. In *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, pages 437–443. Princeton University, 1971.
- [13] Gong Li and Benjamin Watson. Semiautomatic Simplification. In *Symposium on Interactive 3D Graphics*, pages 43–48, 2001.
- [14] David P. Luebke. A Developer’s Survey of Polygonal Simplification Algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.
- [15] M. Macedonia, M. Zyda, D. Pratt, P. Barham, and S. Zeswitz. NPSNET: A Network Software Architecture for Large-Scale Virtual Environments. *Presence*, 3(4):265–287, 1994.
- [16] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono. A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces. In *Computer Graphics International*, pages 180–187, 1999.
- [17] Ryutarou Ohbuchi, Shigeo Takahashi, Takahiko Miyazawa, and Akio Mukaiyama. Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain. In B. Watson and J. W. Buchanan, editors, *Proceedings of Graphics Interface 2001*, pages 9–18, 2001.
- [18] Sylvia L. Osborn, Ravi S. Sandhu, and Qamar Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *Information and System Security*, 3(2):85–106, 2000.
- [19] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust mesh watermarking. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1999.
- [20] Ravi Sandhu, Bhamidipati, and Qamar Munawer. The ARBAC97 Model for Role-Based Administration of Roles. *ACM Transactions on Information and Systems Security*, 2(1), 1999.
- [21] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.
- [22] Ravi S. Sandhu and Pierrangela Samarati. Access Control: Principles and Practice. *IEEE Communications*, 32(9):40–48, 1994.
- [23] N. Shyamsundar and Rajit Gadh. Internet-based Collaborative Product Design with Assembly Features and Virtual Design Spaces. *CAD*, 33:637–651, 2001.
- [24] William Stallings. *Network and Internetwork Security*. IEEE Press, 1995.