Role-Based Viewing Envelopes for Information Protection in Collaborative Modeling

Christopher D. Cera [a] Taeseong Kim [b] JungHyun Han [b]
William C. Regli [a]

[a] *Geometric and Intelligent Computing Laboratory*
Department of Computer Science, College of Engineering
Drexel University
Philadelphia, PA 19104, USA

[b] *Computer Graphics Laboratory*
School of Information and Communications Engineering
SungKyunKwan University
Suwon, 440-746, Korea

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 IA within collaborative design. It is based on a technique we call role-based viewing, which is achieved through integration of multi-resolution geometry and security models. In this way, 3D models are geometrically partitioned, and the partitioning is used to create multi-resolution mesh hierarchies that obscure, obfuscate, or remove sensitive material from the view of users without appropriate permissions. This approach is the basis for our prototype system **FACADE** (the Framework for Access-control in Computer-Aided Design Environments), a synchronous multi-user collaborative modeling environment. In **FACADE**, groups of users work in a shared 3D modeling environment in which each user's viewing and modeling privileges are managed by a central access control mechanism. In this manner, individual users see only the data they are allowed to see, at the level of detail they are permitted to see it.

Key words: Collaborative/distributed design, Access control, Multi-resolution modeling, Role-based viewing

---

Preprint submitted to Elsevier Science

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 [1]. In collaborative design, IA is mission-critical. Suppose a team of designers and subcontractors are working collaboratively on an assembly model. Each has a different set of privileges regarding which aspects of the model they can see and operate on. Further, it may be the case that no individual on the team may have the "need to know" the details of the entire design. This kind of collaboration is common in modern design and manufacturing supply chains, in which designers must interface with others' components, but do so in a way that provides each designer with only the minimal level of information he or she requires to get the task done. 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.

These are all specific instances of IA problems in the context of collaborative design. The authors believe that IA represents a new problem that needs to be addressed in the development of collaborative CAD systems. The authors envision several scenarios in which the work presented in this paper can have impact:

Protect sensitive information: As noted above, designers may have "need to know" rights based on legal, intellectual property, or national security requirements.

Enable 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.

Facilitate 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.

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. Our approach is based on an integration of ideas from IA, feature-based design, multi-resolution modeling and collaborative CAD. This paper addresses the *access control* problem with a combination of *multi-resolution geometry* and *access*

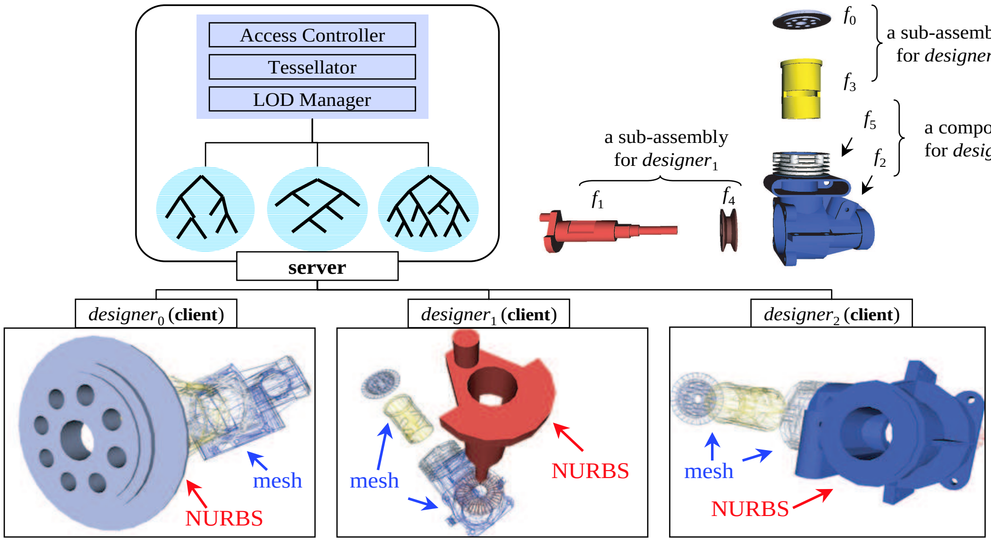


Fig. 1. Secure Collaborative Design System Architecture

control models. Specifically, we introduce:

A 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-centric 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.

Role-based view generation: Given an actor and his/her access authorization, 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 prototype system for role-based collaborative design, FACADE (Framework for Access-control in Computer-Aided Design Environments). In FACADE:

- An assembly model consists of a set of component parts, possibly grouped into sub-assemblies.
- Component parts are represented by and modeled with NURBS [1].
- Design is performed collaboratively by engineers working on different, possibly geographically distant, workstations. FACADE uses a client-server architecture, where the *collaborative CAD server* maintains and synchronizes the master design model. Individual designers work on different sets of components locally, at

---

[1] In the present FACADE, models need not be created from scratch. Pre-existing models from other systems can be imported in a number of CAD (SAT, STEP) and mesh formats (VRML, STL, SMF). Once inside FACADE, they can be edited or manipulated.

their *collaborative CAD clients*.

- The *collaborative CAD server* manages access rights for the users, controlling what they see on their client workstations and what modeling operations are possible. For example, a designer working on a part for which he has write-access would receive a full-resolution NURBS-based model for that particular part. Other parts would be presented in appropriately reduced resolutions, which we call *envelopes*. An envelope can be a convex hull, a bounding box, or a polygon mesh. Both of convex hulls and bounding boxes are easy to create and manage. This paper focuses on meshes. The server tessellates the master CAD model into polygon meshes and creates *multi-resolution* mesh hierarchies to generate the *role-based view* depending on the user's access privileges.
- 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 [3–6]. 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 consists 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) [7–11] have been developed for real-time interactions between distributed collaborators in a number of different domains. Immersive environments such as CAVE [12] have been developed which also support real-time interaction, but they do not necessarily support collaborative CAD. Conner et al. [13] 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. The DOME [14,15] and FIPER [16] systems target the integration of software products, and coordination between them over the network, for collaboration among individuals assigned disjoint duties in the product development cycle or across institutional boundaries. These systems support an access-control framework, but do not offer alternatives to the problem of "all-or-nothing" feature suppression when a lack of full permissions exists. This point also applies to current PDM systems (e.g. TeamCenter, Windchill, and ENOVIA). Lastly, the authors have developed two collaborative design systems, one focusing on group design knowledge capture [17–19] and a second focusing on synchronous authoring of design semantics [20,21].

The FACADE approach combines elements of both the data centric and interaction centric approaches. In this way, FACADE presents a new way of integrating ideas from collaborative graphics with those from collaborative work and engineering design.

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 the CAD domain, information assurance research has been partially addressed through the development of 3D digital watermarking [22–24]. 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 [25]. A component (or a subassembly) 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 simple envelopes.

The Nelsis CAD Framework implemented an access control policy, but the implementation did not go beyond role specification at the project level [26]. The ADOS-X system dealt with coordination between two firms and derived a new access control policy, but this framework was exclusively concerned with controlling access of entire drawings or documents [27]. The problem of authoring geometry and generating “role-based views” among collaborating designers is still unaddressed.

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 recent survey on mesh simplification can be found in [28].

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. See Figure 2.

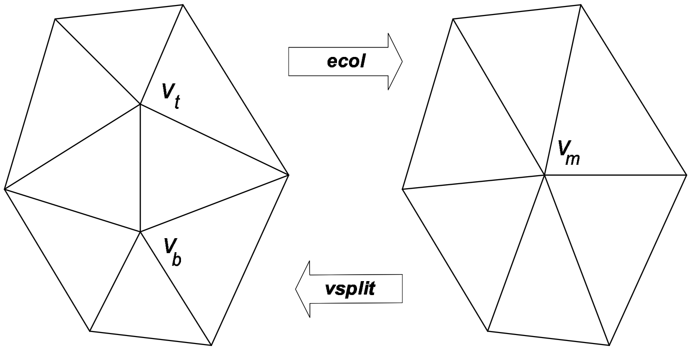


Fig. 2. Illustration of the edge collapse transformation

Vertex split or simply *vsplit* is the inverse operation of *ecol*. Hoppe proposed *progressive mesh*(PM) [29], 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, real-time rendering, and streaming 3D data over the networks. The authors believe that the FACADE prototype is the first system to use these graphics techniques to create a multi-user, multi-security layer, synchronous design environment.

3 Overall Approach: Role-based Viewing for Multi-user Collaboration

In *role-based viewing*, each user sees a shared 3D assembly model in which the constituent components (and their sub-features) are displayed with varying resolutions, determined by the user's *role*. If a user has write permissions to a component, the user receives an editable, NURBS-based CAD model. The other components, where the user might only need to see certain features (or nothing at all), are obfuscated by degrading their visual resolution accuracy to hide the relevant details. The following subsections present the technical development of our framework for role-based viewing in the context of collaborative CAD.

3.1 Access Control Policies

Existing *access control policies* are briefly noted in this subsection. Access control policies commonly found in contemporary systems can be classified as follows [30].

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

Discretionary Access Control (DAC) was originally introduced by Lampson [31], 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.

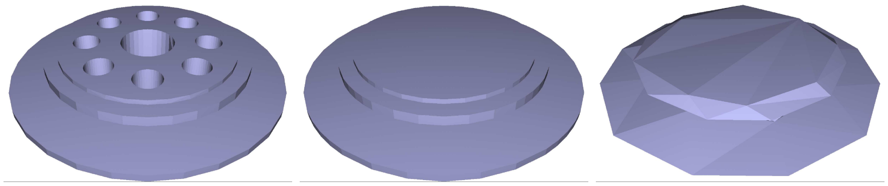
Mandatory Access Control (MAC) [32] policies control dissemination of information by associating users and objects with *security levels*. The security level associated with an object reflects the *sensitivity* of the information, i.e. the potential damage that could result from unauthorized disclosure 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 [30].

In Role-Based Access Control (RBAC) [33], 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).



(a) Original model (for $designer_0$)

(b) Genus-reduced model (for $designer_1$)

(c) Simplified mesh model (for $designer_2$)

Fig. 3. 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 3 shows three different resolutions or LODs of f_0. Figure 3-(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 3-(b) is presented to $designer_1$. Suppose that $designer_2$ is a supplier from another organization. Then, the model in Figure 3-(b) can be again simplified to generate the crude model in Figure 3-(c), which just presents the outline of f_0 to $designer_2$. Those are examples of role-based views. Note that our FACADE system, as based on this framework, provides an appropriate resolution to each designer according to the designer's roles.

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

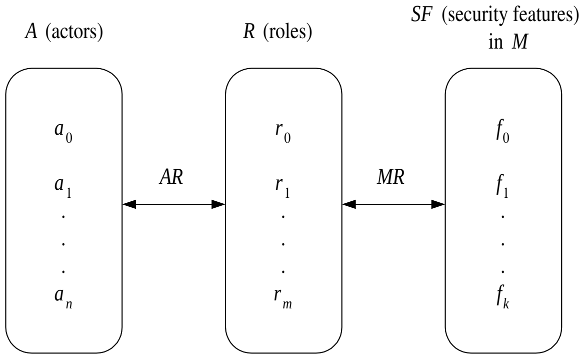


Fig. 4. Actors, Roles and Features

The engineers (designers, process engineers, project supervisors, etc.) correspond to a set of *actors* $A = \{a_0, a_1, \ldots, a_n\}$, 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 4.

The entire assembly design is represented as set of solid models of individual assembly parts, M. A collaborative engineering environment enables multiple engineers (actors) to simultaneously work with M. Let $b(M)$ represent the boundaries of the part models in 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 will use *security features*. Each assembly is described by a set of *security features*, $SF = \{f_0, f_1, \ldots, f_k\}$, where each f_i is a topologically connected point set on $b(M)$ and $\bigcup SF = b(M)$. 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:

$MR \subseteq SF \times R$ (Figure 4).

Example: Suppose that 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 partitions can be created 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

An access matrix is a popular representation 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 (ACLs) 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

Fig. 5. 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 under different role-based views). We developed a matrix implementation as illustrated in Figure 5, 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 instance of an RBAC implementation. To embody a MAC policy in it, we 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 and role-based viewing.

A typical scenario for this RBAC+MAC framework 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.

Example: The administrator not only constructs the access matrix and registers it into an authorization database, but also performs the Actor-Role Assignment *AR*. Suppose that, in the simple example of Figures 1 and 5, actors $designer_0$, $designer_1$, and $designer_2$ are assigned to roles r_0, r_1, and r_2 respectively. Looking at 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 3-(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 3-(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 3-(c). Such a "continuous" role-based viewing technique is discussed in Section 4.

3.4 Multi-user Collaboration

During collaboration, users can take and relinquish control of objects; create and modify existing access privileges for the design; and import and export design session data. There will be a single role-based view generated for each set of actors assigned a common role. Role-based views will be re-generated after each design operation that changes the geometry of the model.

Managing concurrent modeling issues has long been studied by the database community. For example, in a seminal and highly influential paper, Korth et al. [34] adopt an atomic transaction-based approach to synchronizing user changes. The current generation of commercial CAD systems have also attempted to resolve concurrency issues. For example, Parametric Technologies Corporation uses a "token-based" concurrency resolver [35], in which only a single designer can save the session at a time, then finally propagating changes to other users. We provide design conflict alternatives, as outlined by Sun in [36], whenever a conflict arises between multiple actors. Secondly, our MAC policy allows new design changes to be unobtrusively transmitted to other users in the session.

4 Generating Role-Based Views

To an actor a, role-based viewing presents the actor with a new assembly 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 (i.e. they get a full NURBS-based CAD model to work on); if not, the features must be obfuscated so as to hide from a what a does not have permissions to see or modify (e.g. to hide proprietary components from a sub-contractor).

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 versus 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 versus dynamic/continuous: Static simplification usually computes LODs off-line 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

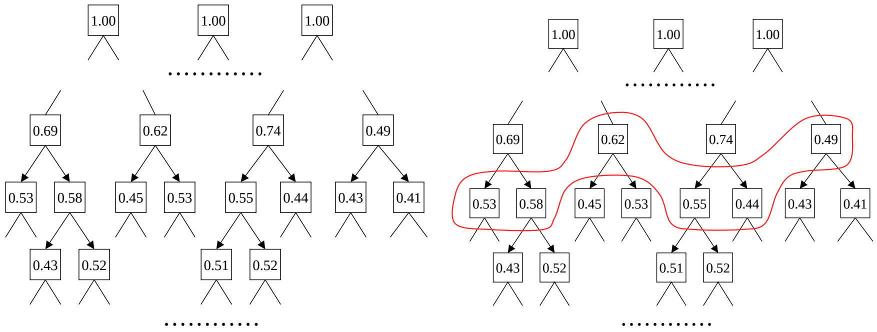
A problem of PM is that it assumes manifold topology, and consequently is not compatible with topology-modifying simplification. Its solution can be found by utilizing *feature-based design* capabilities, which most of contemporary CAD systems support.

Let us consider feature-based 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 [37]. Figure 3-(b) shows a model with negative features removed, and Figure 3-(c) shows the result of applying mesh simplification to the model in Figure 3-(b).

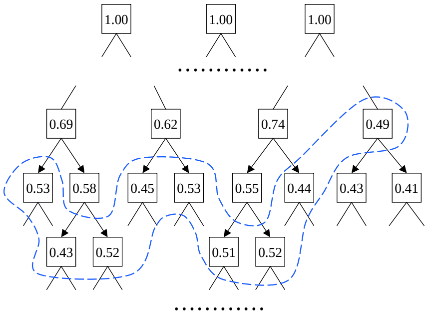
4.3 Role-based Viewing integrated with MAC

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.

We have a PM per a security feature. As discussed in Section 2.3, a PM data struc-



(a) Vertex hierarchy (b) The vertex front with α=0.64



(c) The vertex front with α=0.54

Fig. 6. Progressive Mesh Hierarchy

ture 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 6-(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” [38] 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 6-(b) and -(c). The solution to the task requires understanding of the mesh simplification method we adopted.

Garland and Heckbert [39] 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 *ecol*s are selected basically in order of increasing errors, the inverse operations *vsplit*s 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 6. (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

Observe that, as the second metric 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 6-(b) shows the vertex front determined by α=0.64, and Figure 6-(c) by α=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 α=0.64. Therefore the vertex front of α=0.64 lies higher than that of α=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.

Note that two metrics are required 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 α=0. For that purpose, we subtract 1 from $l_f - l_r$ to set α=$(l_f - l_r - 1)/(l_{max} - l_{min})$ 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 α=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, FACADE, has been developed using OpenGL on Solaris 2.7-2.8 and Windows, and using Mesa and FireGL/GeForce's OpenGL drivers on Linux operating systems.

A designer authenticates with the server to begin a design session where he or she loads pre-existing models, or joins an existing multi-user session. The designer's role association is retrieved, and a *role-based view* is generated using envelopes. An envelope can be a convex hull, a bounding box, or a polygon mesh out of various levels of detail. The multi-resolution envelopes we provide increase the overall security of the system and reduce aggregate bandwidth for communications.

The goals and constraints of the collaboration will dictate how comprehensive the administration requirements should be. The simple authentication mechanism we have created allows administrators to specify both AR and MR assignments. For various administrative configurations, readers are referred to Sandhu [40].

The environment we developed is divided into two stages: authoring and design. The authoring stage is to define AR and MR assignments. An administrator can switch between the two stages. In the authoring stage, the author will see precisely what an actor of the role being specified would see.

Our system supports "fat-clients" where each client maintains a view-independent model. For real-time collaborative design, it would be unacceptable for the server to compute views for each client, for example, whenever a simple affine transformation occurs. The fat-client approach increases the necessary bandwidth requirements when a new user begins or joins a session, but reduces the aggregate bandwidth during the lifetime of the session.

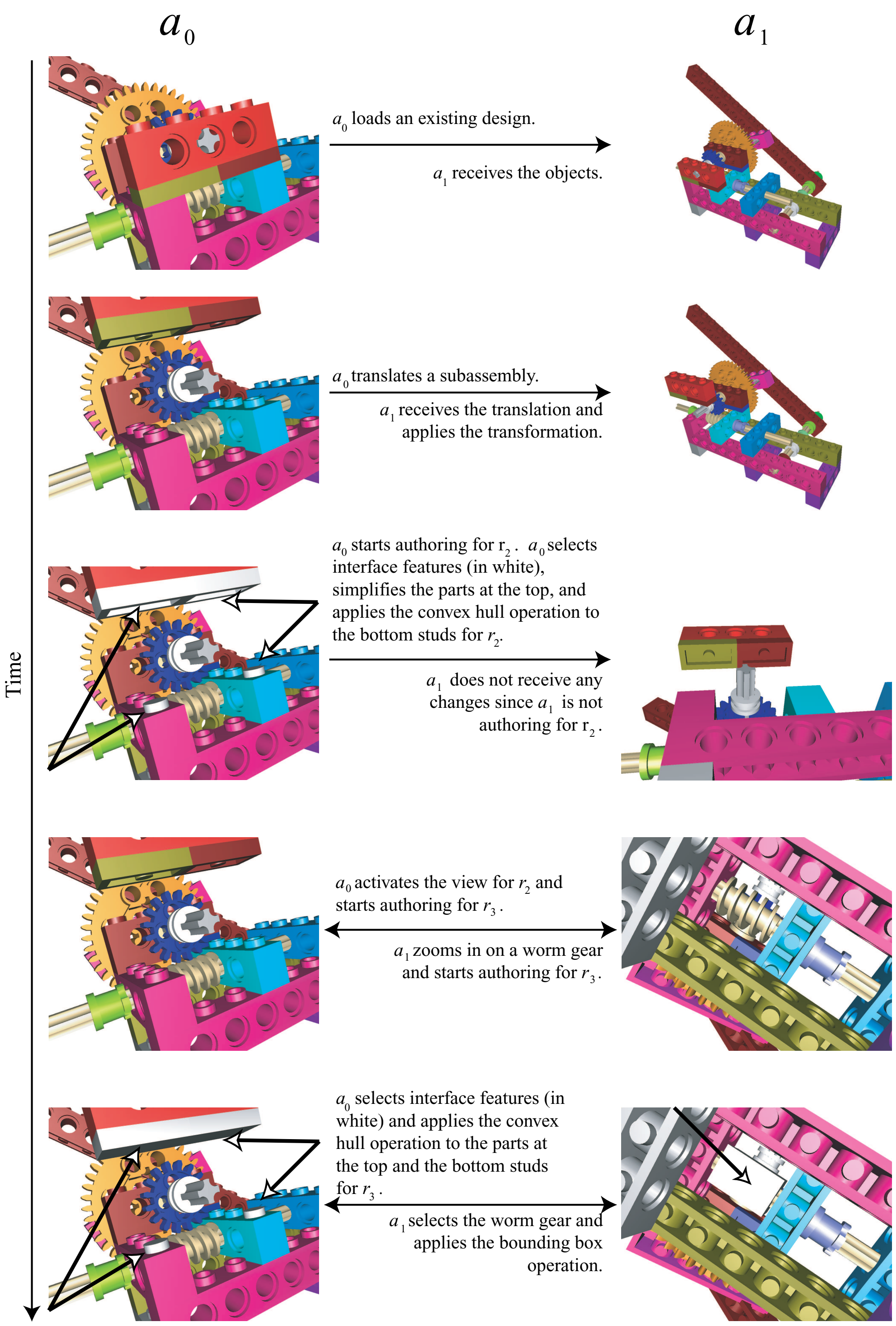
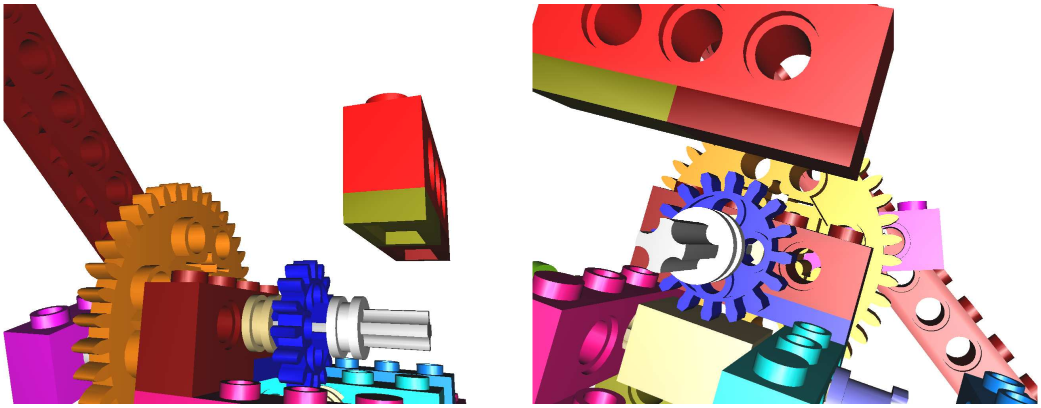


Fig. 7. Authoring of Role-based Views.

A Simple Multi-user Example: Figure 7 gives a storyboard of the role-based authoring process for a simplified windshield wiper assembly [41] designed with Lego^{TM} components. There are two actors (a_0 and a_1), and both of them have non-conflicting roles (r_0 and r_1 respectively). These actors have permissions to author the *MR* assignments of other roles r_2 and r_3, to which actors a_2 and a_3 are assigned respectively.



(a) Role-based view for a_2

(b) Role-based view for a_3

Fig. 8. Role-based Views.

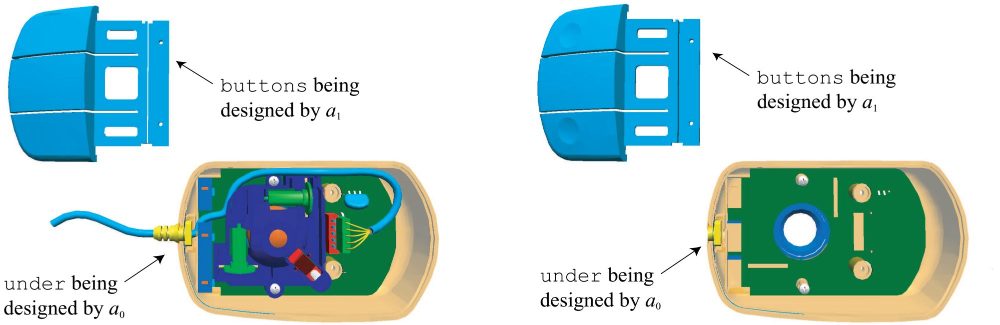
Figure 8 gives the role-based views for actors a_2 and a_3. The view for a_2, depicted in Figure 8(a), is available once a_0 activates it. a_2 might be connected to this session in real-time, or the model will be available from the repository at some later time. Figure 8(b) gives the view for a_3, which includes the pruned features specified by both actors a_0 and a_1.

Example: Mouse Assembly Figure 9 shows two role-based views of a mouse. Figures 9-(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 9-(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. 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 simplified just enough to hide the depressions presented to a_0 in Figure 9-(a).

In contrast, Figure 9-(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.

Example: Motorcycle Engine Assembly A team of engineers are designing the engine assembly given in Figure 10. The team consists of a supervisor a_0, an outsourced engineer a_1 that manufactures the `engine block`, and an outsourced



(a) Role-based view 1 (for a_0) (b) Role-based view 2 (for a_1)

Fig. 9. Test Results for a Mouse Assembly

engineer a_2 in charge of the left and right chambers (i.e. all except the `engine block`). The supervisor has full permissions to view the entire model and the ability to author role information on a "need to know" basis. Engineer a_1 is in charge of casting and machining the `engine block`. The `engine block` interacts with the `crankshaft`, and a_1 has some "need to know" rights to the internal `crankshaft`. However, the `crankshaft` design is proprietary, and therefore the details of the `crankshaft` should not be disclosed to a_1.

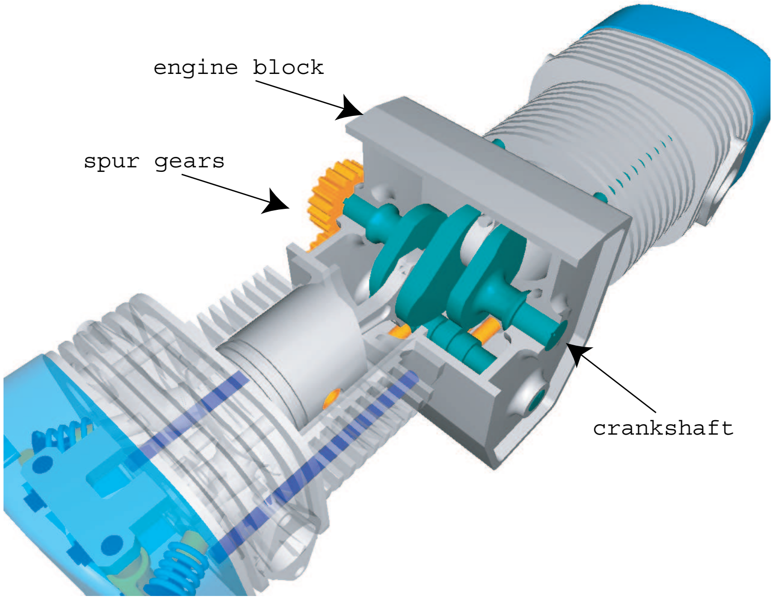
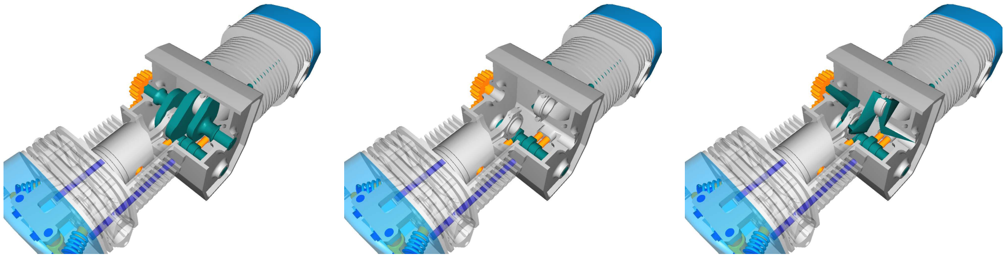


Fig. 10. Motorcycle Engine Assembly

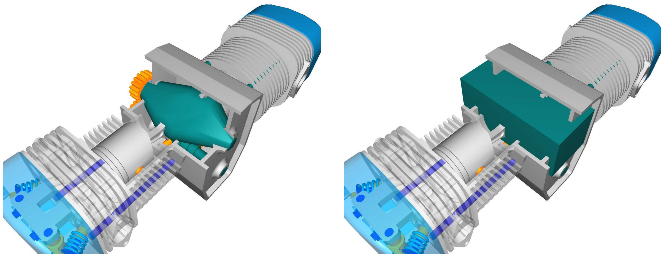
Figure 11 shows four different role-based views that can be provided to a_1, given the original model in Figure 11(a). Figure 11(b) shows the `crankshaft` completely removed from the view. In traditional collaborative design, this might be precisely how this situation would be handled. In role-based viewing, however, we have more options: the object can be tessellated/triangulated and then simplified as in Figure 11(c); the convex hull can be transmitted as in Figure 11(d); the bounding box, as depicted in Figure 11(e), can also be sent.



(a) Original model (as seen by a_0).

(b) Crankshaft removed.

(c) Crankshaft with simplified geometry.



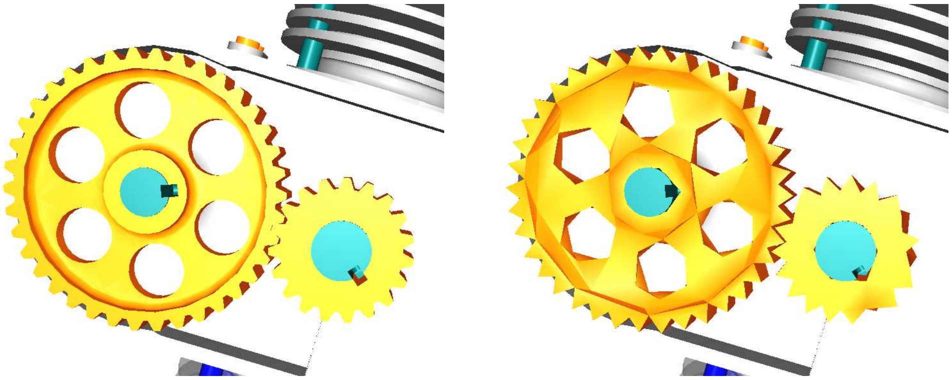
(d) Crankshaft's convex hull.

(e) Crankshaft's bounding box.

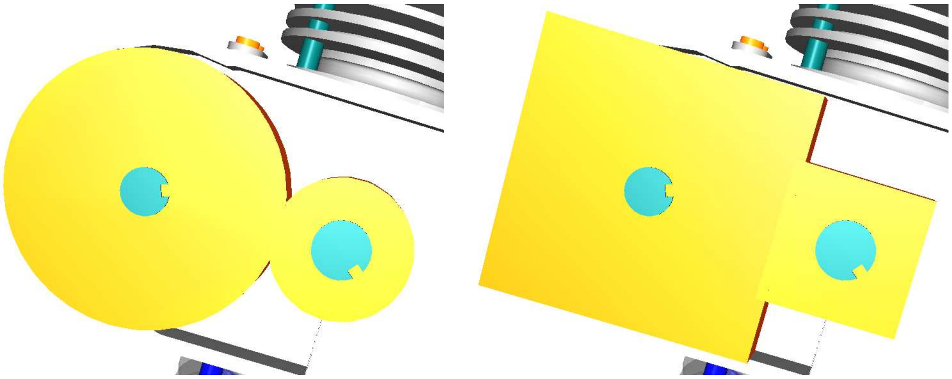
Fig. 11. Role-based View Examples of Engine Assembly

In the engine assembly example, it might be useful to remove or obscure information about the gears. Figure 12 contains the candidate role-based views of the *spur gears* for a_1. The original model is given in Figure 12(a). The simplified model depicted in Figure 12(b) clearly obfuscates many features of the teeth (i.e. addendum, dedendum, clearance, pressure angle, circular tooth thickness, circular pitch, fillet radius, etc.), but retains the overall shape and conveys, for example, that the big gear has 6 holes. The convex hull in Figure 12(c) hides the holes, but still gives the outside diameter. The bounding box in Figure 12(d) has a similar effect as the convex hull, but is less revealing that it is even a gear.

Figure 13 depicts a candidate role-based view for a_2. The alternatives that a_0 can send to a_2 are similar in concept to the last example, except details of the engine block and associated components need not be disclosed. Figure 13 simply gives the convex hull of the engine block.



(a) Original gear. (b) Gear with simplified geometry.



(c) Gear’s convex hull. (d) Gear’s bounding box.

Fig. 12. Role-based Viewing Examples of Exterior Gears of Engine Assembly

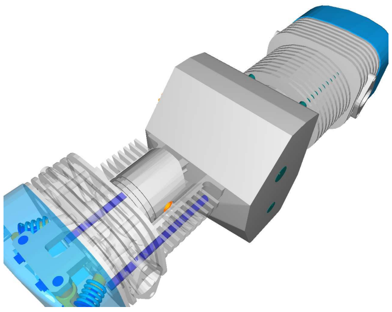


Fig. 13. Role-based Viewing Example of Engine Block

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 embodiment of a MAC policy within an RBAC framework, implemented as an access matrix. Recent works on RBAC proposed sophisticated structures such as role hierarchy [40]. 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 extending the access matrix with a role hierarchy.

We have developed the notion of *security features* and proposed using 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 user-guided simplification [42,43] might need to be employed. User-guided simplification is a means of supervising the mesh reduction process by editing the order of *ecol*s, selecting regions where more or less simplification is necessary, and directly manipulating the vertex hierarchy. One disadvantage of user-guided simplification is that parameters of the simplification will need to be stored with the model, since these cannot be automatically derived.

Our current and future work consists of refinements to the overall system, use of multi-resolution NURBS directly on the models, and integration of knowledge capture and annotation techniques [19,17] to record design rationale and describe the semantics of the structure, behavior and function of the device.

Acknowledgments 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; Office of Naval Research (ONR) Grant N00014-01-1-0618, and Korea Research Foundation Grant KRF-2001-013-E00073. 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] W. Stallings, Network and Internetwork Security, IEEE Press, 1995.

[2] S. Callahan, J. Heisserman, A Product Representation to Support Process Automation, in: M. Pratt, R. Sriram, M. Wozny (Eds.), Product Modeling for Computer Integrated Design and Manufacture, Chapman and Hall, 1996, pp. 285–295.

[3] S. Szykman, R. D. Sriram, W. C. Regli, The role of knowledge in next-generation product development systems, ASME Transactions, the Journal of Computer and Information Science in Engineering 1 (1) (2001) 3–11.

[4] M. R. Cutkosky, J. M. Tenenbaum, J. Glicksman, Madefast: Collaborative engineering over the internet, Communications of the ACM 39 (9) (1996) 78–87.

[5] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, W. R. Swartout, Enabling technology for knowledge sharing, A.I. Magazine 12 (3) (1991) 37–56.

[6] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, J. C. Weber, Pact: An experiment in integrating concurrent engineering systems, IEEE Computer 26 (1) (1993) 28–38.

[7] J. W. Barrus, R. C. Waters, Locales: Supporting Large Multiuser Virtual Environments, IEEE Computer Graphics and Applications 16 (6) (1996) 50–57.

[8] J. M. S. Dias, R. Galli, A. C. Almeida, C. A. C. Bello, J. M. Rebordao, mWorld: A Multiuser 3D Virtual Environment, IEEE Computer Graphics and Applications 17 (2) (1997) 55–65.

[9] H. Eriksson, MBONE: The Multicast Backbone, Communications of the ACM 37 (8) (1994) 54–60.

[10] S. Jayaram, U. Jayaram, Y. Wang, H. Tirumali, K. Lyons, P. Hart, VADE: a Virtual Assembly Design Environment, IEEE Computer Graphics and Applications 19 (6) (1999) 44–50.

[11] M. Macedonia, M. Zyda, D. Pratt, P. Barham, S. Zeswitz, NPSNET: A Network Software Architecture for Large-Scale Virtual Environments, Presence 3 (4) (1994) 265–287.

[12] C. Cruz-Neira, D. J. Sandin, T. 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, ACM Press, 1993, pp. 135–142.

[13] B. Conner, M. Cutts, R. Fish, H. Fuchs, L. Holden, M. Jacobs, B. Loss, L. Markosian, R. Riesenfeld, , G. Turk, An Immersive Tool for Wide-Area Collaborative Design, in: TeamCAD, the First Graphics Visualization, and Usability (GVU) Workshop on Collaborative Design, 1997, pp. 139–143.

[14] F. Pahng, N. Senin, D. Wallace, Distribution Modeling and Evaluation of Product Design Problems, Computer-Aided Design 30 (6) (1998) 411–423.

[15] S. Abrahamson, D. Wallace, N. Senin, P. Sferro, Integrated Design in a Service Marketplace, Computer-Aided Design 32 (2) (2000) 97–107.

[16] P. J. Rohl, R. M. Kolonay, R. K. Irani, M. Sobolewski, K. Kao, M. W. Bailey, A Federated Intelligent Product Environment, in: Proceedings of the Eighth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2000.

[17] S. V. Lombeyda, W. C. Regli, Conceptual Design for Mechatronic Assemblies, in: Proceedings of the Fifth ACM symposium on Solid modeling and applications, ACM Press, 1999, pp. 320–321.

[18] S. Lombeyda, W. C. Regli, Conceptual design for assembly, in: S. K. Gupta (Ed.), ASME Design Engineering Technical Conferences, Fourth Design for Manufacturing Conference, ASME, ASME Press, New York, NY, USA, 1999, las Vegas, NV.

[19] L. P. Anthony, J. E. John, S. V. Lombeyda, W. C. Regli, Cup: A computer-aided conceptual design environment for assembly modeling, ASME/ACM Transactions, The Journal of Computer and Information Science in Engineering 1 (2) (2001) 186–192.

[20] C. D. Cera, W. C. Regli, I. Braude, Y. Shapirstein, C. V. Foster, A collaborative 3d environment for authoring design semantics, IEEE Computer Graphics and Applications 22 (3) (2002) 43–55.

[21] C. V. Foster, Y. Shapirshteyn, C. D. Cera, W. C. Regli, Multi-user modeling of nurbs-based objects, in: ASME Design Engineering Technical Conferences, Computers and Information in Engineering Conference (DETC 2001/CIE-21256), ASME Press, 2001.

[22] R. Ohbuchi, H. Masuda, M. Aono, A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces, in: Computer Graphics International, 1999, pp. 180–187.

[23] R. Ohbuchi, S. Takahashi, T. Miyazawa, A. Mukaiyama, Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain, in: B. Watson, J. W. Buchanan (Eds.), Proceedings of Graphics Interface 2001, 2001, pp. 9–18.

[24] E. Praun, H. Hoppe, A. Finkelstein, Robust Mesh Watermarking, in: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Pub., 1999, pp. 49–56.

[25] N. Shyamsundar, R. Gadh, Internet-based Collaborative Product Design with Assembly Features and Virtual Design Spaces, Computer-Aided Design 33 (9) (2001) 637–651.

[26] A. J. van der Hoeven, O. ten Bosch, R. van Leuken, P. van der Wolf, A Flexible Access Control Mechanism for CAD Frameworks, in: Proceedings of the conference on European design automation conference, IEEE Computer Society Press, 1994, pp. 188–193.

[27] G. Stevens, V. Wulf, A New Dimension in Access Control: Studying Maintenance Engineering Across Organizational Boundaries, in: Proceedings of the 2002 ACM conference on Computer supported cooperative work, ACM Press, 2002, pp. 196–205.

[28] D. P. Luebke, A Developer's Survey of Polygonal Simplification Algorithms, IEEE Computer Graphics and Applications 21 (3) (2001) 24–35.

[29] H. Hoppe, Progressive Meshes, in: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM Press, 1996, pp. 99–108.

[30] R. S. Sandhu, P. Samarati, Access Control: Principles and Practice, IEEE Communications 32 (9) (1994) 40–48.

[31] B. Lampson, Protection, in: Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems, Princeton University, 1971, pp. 437–443.

[32] S. L. Osborn, R. S. Sandhu, Q. Munawer, Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies, Information and System Security 3 (2) (2000) 85–106.

[33] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, Role-Based Access Control Models, IEEE Computer 29 (2) (1996) 38–47.

[34] F. Bancilhon, W. Kim, H. Korth, A model of CAD transactions, in: Proceedings of the Conference on Very Large Databases (VLDB), 1985, pp. 25–33.

[35] http://www.ptc.com.

[36] C. Sun, D. Chen, Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems, ACM Transactions on Computer-Human Interaction 9 (1) (2002) 1–41.

[37] J. El-Sana, A. Varshney, Controlled Simplification of Genus for Polygonal Models, in: IEEE Visualization, 1997, pp. 403–412.

[38] H. Hoppe, View-Dependent Refinement of Progressive Meshes, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 189–198.

[39] M. Garland, P. S. Heckbert, Surface Simplification Using Quadric Error Metrics, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216.

[40] R. Sandhu, V. Bhamidipati, Q. Munawer, The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and System Security (TISSEC) 2 (1) (1999) 105–135.

[41] D. Subramanian, C. Wang, Kinematic synthesis with configuration spaces, in: Proceedings of Qualitative Reasoning 93, 1993, pp. 228–239.
URL
`http://citeseer.nj.nec.com/subramanian93kinematic.html`

[42] Y. Kho, M. Garland, User-guided Simplification, in: Proceedings of the 2003 symposium on Interactive 3D graphics, ACM Press, 2003, pp. 123–126.

[43] G. Li, B. Watson, Semiautomatic Simplification, in: Symposium on Interactive 3D Graphics, 2001, pp. 43–48.