A Collaborative 3D Environment for Authoring of Design Semantics

Christopher D. Cera
William C. Regli
Ilya Braude
Yuri Shapirstein
and
Cheryl V. Foster

Technical Report DU-CS-01-06
Department of Computer Science
Drexel University
Philadelphia, PA 19104
September 2001

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2001		**00-09-2001 to 00-09-2001**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Collaborative 3D Environment for Authoring of Design Semantics	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science,Drexel University,Philadelphia,PA,19104**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**16**	

A Collaborative 3D Environment for Authoring of Design Semantics

Christopher D. Cera* William C. Regli† Ilya Braude‡ Yuri Shapirstein§ Cheryl V. Foster¶

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
http://gicl.mcs.drexel.edu

Abstract

In this paper we present **MUG**, a collaborative 3D environment to support knowledge-based conceptual design. This work integrates Computer-Aided and Collaborative Design methods with representation framework being developed for the Semantic Web. In this work, we have created a unique environment that allows a group of users, working together over the Internet, to create both a 3D layout and a knowledge-based description of a conceptual product design. MUG accepts domain-specific engineering ontologies described with the DARPA Agent Markup Language (DAML) and provides a multi-modal collaborative 3D environment, supporting shared audio and shared 3D manipulation, for users to define product structures. MUG is written entirely in the Java language and runs across a wide wide variety of hardware and operating system platforms.

Keywords: Conceptual Design, Computer-Aided Conceptual Design, Computer-Aided Design, Engineering Ontologies.

1 Introduction

Design is a ubiquitous human activity and a critical aspect of many modern industries. Computer-Aided Design (CAD) tools have revolutionized product development in the automotive and aerospace (Mechanical CAD) and Architecture-Engineering-Construction (AEC) industries, as well as in many other areas where geometric computation and visualization have proven essential. Presently, however, design and manufacturing industries from across this spectrum have been shifting toward collaborative environments that are distributed and both knowledge and geometrically intensive.

It is becoming increasingly common for design teams to be separated geographically. While the existing generation of CAD environments and product data management systems provide an excellent set of tools for individual authoring of geometry, topology and constraints and database synchronization of these entities, they do not as yet support true collaborative authoring of *design semantics*. Design semantics, e.g., grounded representations of product and process knowledge, are growing to be vitally important for creating the next generation of knowledge-intensive computer-aided design applications [14].

This paper presents an approach to collaborative authoring of design semantics within multi-user 3D environment. Our research unites ideas from traditional engineering design with recent work from the Semantic Web and knowledge engineering community. The system we present, **MUG**, allows a team of designers, collaborating over the Internet in a shared, multi-modal, 3D workspace, to describe both a 3D layout and as well as semantically grounded behavioral description of a product or device. Specific aspects of our approach, and the MUG system, include:

*Email: cera@drexel.edu.
†URL: http://gicl.mcs.drexel.edu/regli; Email: regli@drexel.edu.
‡Email: ib29@mcs.drexel.edu.
§Email: uyshapir@mcs.drexel.edu.
¶Email: ucfoster@mcs.drexel.edu.

- **Integration of Functional Modeling with Sketch-based Conceptual Design.** Previous work on conceptual design tools can be loosely classified into two categories. From the mechanical engineering community, tools have been developed to describe designs as a graph of functional relationships; from the graphics and modeling community, tools have been created to turn "back-of-the-envelope" sketches into 3D designs and solid models.

  The MUG approach draws on both of these, creating a merger which users perform controlled sketching of the design's layout and, in doing so, capture the functional representation.

- **Group Authoring of Design Semantics.** Modeling device function in MUG requires having a representation of the design's functional semantics. Many fields in industry are demanding for more powerful, and extensible representation formats for sharing product data (e.g., eXtensible Markup Language (XML) is currently receiving considerable industry attention) but most existing approaches are merely shared syntax.

  With MUG, users can describe device behavior and function using ontologies defined in the DARPA Agent Markup Language (DAML). DAML is an emerging W3C-backed representation for the Semantic Web; we use it to show how users can create logically grounded ontologies that are domain-specific, and use them to annotate designs created with MUG. Further, MUG also uses DAML as its protocol for all peer-to-peer and peer-to-sever communication.

- **Multi-modal User Collaboration.** The MUG prototype provides several modalities for interaction. First, and most importantly, users can control and model individual shapes in the workspace. One example is of multiple users, operating at a distance, working shaping the same NURBS-based surface. Secondly, MUG also includes a shared, multi-cast audio communication channel. Designers can work and *converse* in the digital design space.

- **Shared, *n*-user 3D Workspace.** Lastly, MUG uses Java, Java3D and JavaSpaces to build a shared 3D workspace. Shared realities have been extensively studied by the virtual reality and graphics communities for some time. Our main contribution to this extensive discussion is practical one: showing, by proof-of-concept, that a fairly sophisticated, multi-platform collaborative modeling environment can be created entirely in Java and interactively handle CAD models of surprisingly high complexity.

In this way, MUG integrates collaborative CAD with several current trends in information technology from the areas of product data representation and product knowledge sharing with Java, Java 3D and the Semantic Web effort. MUG creates a structured way in which product data semantics can be described as designs evolve. We will describe two scenarios for using MUG: first, designers can create conceptual designs for new products. In to support this scenario, MUG provides a set of primitive shapes and tools for defining component relationships to enable a group of users to create "back-of-the-envelope" sketch of a new device. The second way to use MUG is as a collaborative knowledge annotator for describing the behavioral semantics of shapes. We provide a detailed example, a one degree of freedom scanner, and show how it's behavior can be defined with MUG. We see MUG as an extension of computer support to the very early phases of product development, as well as a means of improving interfaces between downstream CAx tools and applications.

Paper Organization. This paper is organized as follows: Section 2 describes related work from traditional Conceptual Design and Computer-Aided Design. Section 3 presents our approach to conceptual design, relating the process to the "bottom-up" construction of a knowledge-base and the creation of shared semantics among designers. Sections 4 describes the MUG system, some of our software design issues and the use of the DARPA Agent Markup Language (DAML) to capture shared design semantics. Section 5 provides an example of how a tool like MUG would be used during design. Lastly, Sections 7 and 8 discusses and summarizes our results, presents our conclusions, and outlines goals for future research.

2 Related Research

Broadly speaking, existing approaches (and software tools) for early stage design, fall into one of two categories: those based on *functional modeling* and those based on *sketching and layout*. The *functional modeling* approach finds its roots in traditional mechanical design, described above. *Sketch-based* conceptual design [7, 9, 5, 4, 11, 3, 6] incorporates ideas from computer graphics, user interface design and computer vision in order to facilitate the speedy creation of detailed 3D designs, usually from elementary 2D sketches. The goal of both methods is to improve support for the flexible generation of alternative design ideas at the early stages. The conceptual designs can then be considered for aesthetics, or used for engineering analysis, cost estimation or any number of other preliminary design assessments.

The *sketch-based* approaches to conceptual design cover a diverse collection of domains. *Conceptual sketching* has been explored as a way to develop user interfaces [5], architectural drawings [9] and 3D solid models for CAD [3, 9, 7, 11, 4, 6]. One approach is to make these systems *feature-based*, where domain-specific features can be exploited to drive the recognition and refinement of the conceptual design into a detailed 3D model [5].

Many of the core ideas used in the *functional modeling* approach to conceptual design have been used in the mainstream CAD and design research communities to develop software environments to support conceptual design. Much of the work in this area comes from academia as research in *Computer-Aided Conceptual Design* (CACD), and has lead to several prototype systems. These systems most often take a graphical (nodes and edges) [10, 1] approach to describing the engineering relationships among the elements in a mechanical assembly model, or other CAD-based structure [8].

3 Approach

The approach that we take to conceptual design unites functional modeling with elements of the sketch-based methods and with ideas from Computer Supported Cooperative Work (CSCW). Our observation is that one can view the design process as a problem in creation of shared design semantics. In the past, shared semantics were usually imposed, *a priori*, before the development could begin, e.g., firms used a single turn-key CAD environment. The need for common knowledge-sharing standards (which go beyond data standards such as STEP) has been often cited as a major obstacle to achieving true concurrent engineering.

MUG takes an analogous approach to conceptual design: a team of designers engage in a collaborative problem solving exercise in order to define the semantics of a new product. Within MUG the conceptual design process is one in which designers collaboratively author an assembly design, specifying its structure and layout in 3D while annotating the relationships among all the entities in the assembly with reference to base ontologies. For a mechanical design application, the upper-level ontologies used describe fundamental structure-behavior-function properties common in mechanical systems. Hence, as designers create their 3D "back of the envelope" sketch of their design, they also are authoring the set of logical sentences that describe the semantics of the assembly with reference to these base ontologies. To modify MUG to perform conceptual design for the Architecture/Engineering/Construction (AEC) domain, one would only need to provide the suitable 3D conceptual design primitives and references to the appropriate base ontologies for describing relationships in the AEC world. This approach is similar to that favored by the knowledge engineering community for development of large, shared knowledge-bases.

3.1 Conceptual Description of Design Structure

Designers often model preliminary designs without having detailed specifications for every component and feature, being more concerned with specifying design intent and the inter-relationships between the components and features. During detailed CAD, design activities capture detailed geometries, assembly constraints and kinematics—but not the logical semantics of the product structure. MUG supports design activities from both of these directions. First, for conceptual design, MUG has elementary CAD capabilities: in 3D, the users define major components, either by placing and scaling Constructive Solid Geometry (CSG) primitives or by importing pre-existing geometries, and arranging the primitives in a layout which makes logical sense. In this way, MUG pushes the users away from the tedious task of modeling the detailed geometry toward the task of modeling the design logical structure. An example of the MUG interface and some of its CSG-like design primitives is shown in Figure 1. Second, MUG can import pre-existing geometries (such as those created in a traditional CAD system) and then be used as a knowledge annotator for describing the assembly semantics. An example of this is shown in Figure 2.

The logical structure of the design is captured as a set of arbitrary relationships among the shape primitives. For example, in the case of electro-mechanical design the emphasis may be on the *function* of the components in the assembly and the *flow* of a substance or material from one or more components to some other. That is, the laying out and defining of the structural, behavioral, and functional relationships in the assembly are more salient at this point in the design process than geometry, size, shape, rotation, even relative orientation. It is the structure-behavior-function knowledge, more than the geometry and topology, that encodes the designers' intent.

MUG represents these as *links* in a graph of the assembly semantics. Users "mark up" the design as they create the 3D layout of physical components and thus capture a semi-formal notion of design intent. For example, why does the `pivot pin` connect to the `motor`? What energy flows from `motor` to the `lens housing`, electrical or hydraulic? The semantics of these queries and their answers can be expressed in terms of the design ontologies used by MUG.

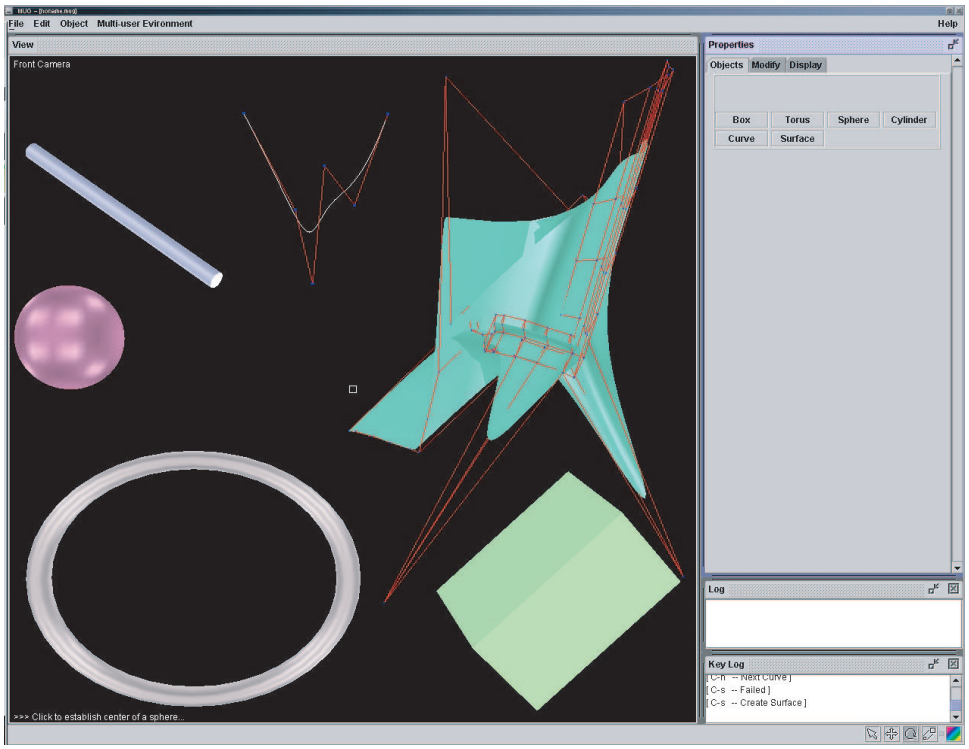


Figure 1: A shot of the MUG interface and some of its design primitives: blocks, tori, spheres, cylinders/frustums and NURBS-based curves and surfaces.

Our running example for this paper, shown in Figure 2, is a conceptual design of a one degree of freedom scanner. This type of electro-mechanical assembly is quite common in devices such as range finders, printers and small-scale sensors. This design has five major components: a base, gimble ring, pivot pin, electrical motor and a housing for the scanning lens or sonar array. Figure 2 (a) shows the model as it might be create in mug, either directly or, as in this case, by import pre-existing geometries from other CAD environments. Figure 2 (b) shows the annotated scanner, with links describing relationships among the major components.

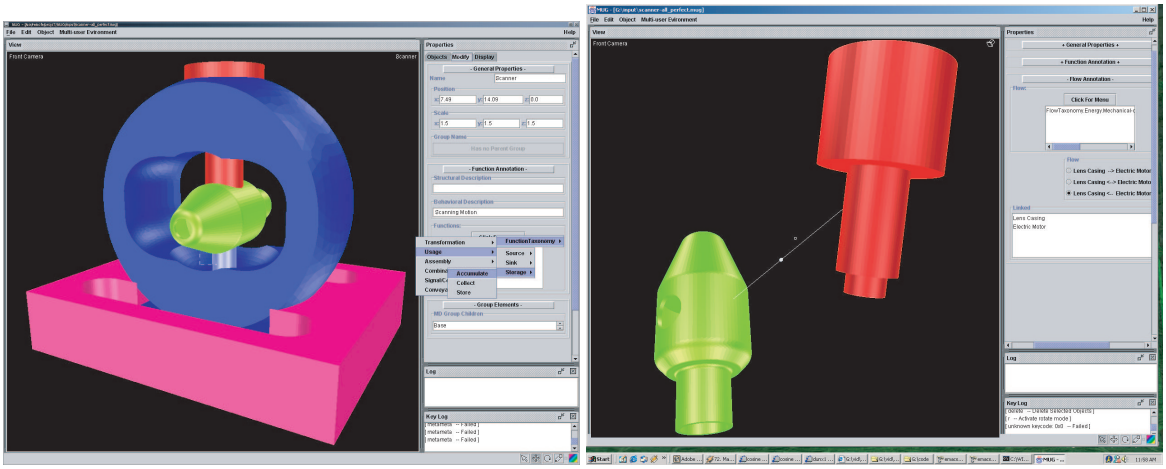
3.2 Description of Design Semantics

The specific design semantics will depend on the engineering domain in which MUG is being used. Regardless of the domain, however, several common patterns exist:

- Domains often have categories and taxonomies of primitive elements. In electro-mechanical CAD for example, fasteners, motors, linkages, etc. all have well-established, catalog-like hierarchies. In AEC, buildings floor plans, structural elements, etc. also have similar taxonomic structures.
- The relationships among the primitive elements is also often defined by some well-established communal vocabulary. For example, electro-mechanical CAD systems model assemblies, subassemblies, and the kinematic connections among these components.

Our present approach is to model these taxonomies, including the attributes and logical properties of each taxonomy element, as a formal Ontology. The Knowledge Representation community has made considerable strides in recent years, most recently in the W3C-backed and DARPA-sponsored Semantic Web[1] effort. We leverage this research and these emerging representations to allow users of MUG to import their own domain ontologies and use this to define the semantics of the 3D layout they create. Our representational syntax is based on the DARPA Agent Markup Language (DAML). In

[1] `http://www.w3c.org; http://www.semanticweb.org; http://www.daml.org.`



(a) A 1-DOF scanner assembly.

(b) Exploded view with functional links.

Figure 2: A design of a 1 degree of freedom scanner loaded into MUG for collaborative annotation.

the following sections, we give a brief background on DAML and describe how we use DAML to describe the semantics of two different engineering domains for use in MUG.

3.2.1 The Role of DARPA Agent Markup Language (DAML)

The DARPA Agent Markup Language (DAML) is the central part of the effort to create a model theoretic and axiomatic semantics for the information on the Internet and World Wide Web. XML has been widely embraced as the new collective syntax for describing data and information, and as a successor to HTML as a the method for describing how web pages should be displayed. The fundamental limitations of XML addressed by DAML include its limited ability to describe inter-object relationships (e.g., taxonomies, schemas and ontologies) and rules of inference for the data described.

DAML provides a set of logical constructs for defining ontologies; these ontologies can be used to rigorously represent knowledge and annotate information so that it can be interpreted by digital, as well as human, agents. The syntax of DAML is based on XML, XML Schema and the Resource Description Framework (RDF). While DAML's initial intent is as a description language for the data provided on the web, our group is actively exploring using it as a general knowledge representation language and information interchange format.

MUG makes use of DAML in three unique ways:

1. The MUG protocols for communication and interaction are defined as DAML messages, i.e., all MUG agents speak DAML to one another.

2. MUG uses DAML to define its native file format, i.e., MUG worlds are saved as DAML files.

3. Lastly, and most significantly, DAML ontologies for different design domains can be dynamically loaded into MUG and used by a collaborating group of designers to perform knowledge markup of the evolving 3D layouts.

Here we provide two example domains to illustrate how different ontologies can be used to annotate collaborative designs in MUG.

3.2.2 Electro-Mechanical Design Semantics

In Electro-Mechanical Design, there are three basic knowledge-level descriptors needed to capture the meaning of components: *function*, i.e., what an object (component, group of components, or subassembly) does within the overall assembly; *behavior*, the observable actions by which it accomplishes this function; and *structure*, specifically what attributes of the physical object (e.g., physical constraints, materials, and so on) are related to helping it achieve this behavior. Function

and flow can be used to label the substances or materials which are the inputs and/or outputs of the object's function and create a graph that captures the object's intended dynamics.

One of the current MUG's ontologies is based on the work of Szykman et al. [13, 12] at the National Institute of Standards and Technology (NIST). They created a unified way of representing data and artifacts using a series of taxonomies or ontological schema trees by which *function* and *flow* can be expressed. These taxonomies had been compiled through a survey of actual engineers' documents and design notebooks regarding representation of design language by engineers. The NIST work can describe functions in multiple domains, such as `Math-Logical/Divide` or `Energy-Sink/Dissipate`; structural connections such as `Enclose`, `Fasten`, or `Guide`; and functional properties such as `Rotational-Movement` or `Translational-Movement`. Components or modules within assemblies can be tagged with elements from these taxonomies to represent their purpose and build a description of the whole unit.

As part of MUG, our team converted the NIST function-flow taxonomy into a DAML ontology. Presently, this ontology includes over 100 functions and over 130 flows. Users can "markup" a particular component, group of components, or link several components with the representational attributes built from this ontology. Embedding structure-behavior-function information in MUG's model files enables the addition of abstract design intent and purpose to the simple concrete objects and groups represented in each design world.

3.2.3 AEC Design

We have also created prototype Architecture/Engineering/Construction (AEC) ontology using information from the Universal Standard Products and Services Classification (UNSPSC)[2]. Starting with UNSPSC, which is a vast classification scheme for global business endorsed by the United Nations, we have created an initial DAML ontology describing building structures, wiring, piping and heating/ventilation/air conditioning (HVAC) connectivity. This ontology has been used by MUG to annotate floor plans in Architectural describes.

3.3 Multi-user and Multi-modal Collaboration

MUG creates a synchronous collaboration space that is shared over the network. During a design session, users can manipulate objects in the shared space and the changes are propagated to the other active users. Concurrently, users can maintain a multi-cast IP-based audio channel for group discussion. In this way, MUG attempts to digitally enable group work in a shared work area, i.e., a 3D digital conference table. During collaboration, users can take and relinquish control of objects; create and modify the ontological annotations for the design; and import and export design session data. Some unique elements of the MUG collaboration space include the ability to record the entire collaborative exchange, both audio discussion and the design process history, and the ability to have fine levels of shared control over different objects in the scene.

Figure 4 gives a brief storyboard showing how two users can work to manipulate the same NURBS surface. **NURBS** (**N**on-**U**niform **R**ational **B**-**S**plines) are a standard representation for complex surfaces and shapes in most commercial CAD packages. The shape of a NURBS form is controlled by four main parameters: the *control points*, *weights*, the *degree* of the curve, and the *knot vector*, (i.e. degrees of freedom) for the curve. MUG enables a group of users to collaboratively manipulate a single NURBS object by allowing them to partition the set of control points, with each user controlling a different set of points. Changes to the shape of the surface are then propagated to other users in the MUG session.

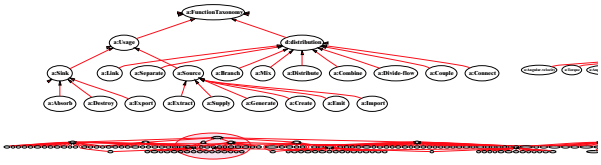
4 MUG System Architecture

There are two major components that make up MUG's multi-user architecture: the **MUG Server** and the **MUG Client**. A separate JavaSpace is used synchronize all communication in the collaborative environment.

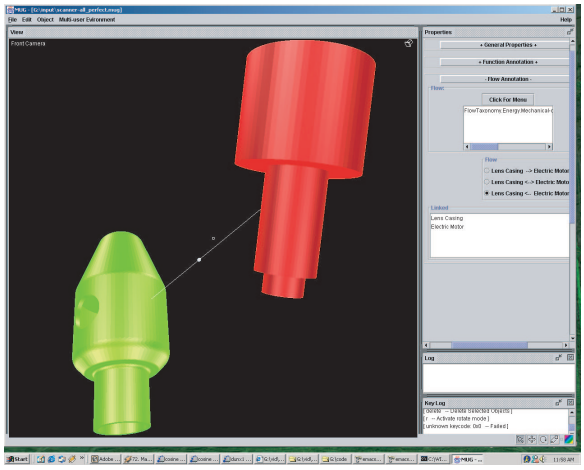
MUG Server. The Server is consists of two main components: **MUG Manager** and the **Design Agent**. The **MUG Manager** is aware of all clients that are currently connected to a particular collaborative design session. This architecture uses a single server process to handle all sessions currently operating, with each session having their own respective vector of clients working within them.

The **Design Agent** is an agent that is started by the server when a client makes a request to initialize a new design space. This agent is the representative authority for communication between the server and all clients within that particular session.

[2] `http://eccma.org/unspsc/`



(a) Our ontology electro-mechanical Function-Flow, with three highlighted portions.



(b) Annotation of Function-Flow within MUG for the motor/lens housing link.

```
<rdf:RDF
    xmlns:rdf  = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
    xmlns:daml = "http://www.daml.org/2001/03/daml+oil#"
    xmlns:xsd  = "http://www.w3.org/2000/10/XMLSchema#"
    xmlns:mug  = "http://edge.mcs.drexel.edu/MUG/2001/05/29/mug#"
    xmlns:sbf  = "http://edge.mcs.drexel.edu/MUG/2001/09/01/sbf#"
>

<mug:World rdf:ID="Scanner">
    ....
    <mug:ObjectHandle>
        <mug:Object rdf:ID="Lens_Casing>
            <mug:Geometry>
                ....
            </mug:Geometry>
            <sbf:Structure></sbf:Structure>
            <sbf:Behavior></sbf:Behavior>
            <sbf:Function rdf:resource="sbf#Translate"/>
        </mug:Object>
    </mug:ObjectHandle>
    ....
    <mug:ObjectHandle>
        <mug:Object rdf:ID="Electric_Motor">
            <mug:Geometry>
                ....
            </mug:Geometry>
            <sbf:Structure></sbf:Structure>
            <sbf:Behavior>"Power_Converter"</sbf:Behavior>
            <sbf:Function rdf:resource="sbf#Convert"/>
        </mug:Object>
    <mug:ObjectHandle>
    <mug:GroupHandle>
    ....
    </mug:GroupHandle>
    <mug:LinkHandle>
        <mug:Link rdf:ID="link1"/>
            <mug:Member rdf:resource="#Lens_Casing/>
            <mug:Member rdf:resource="#Electric_Motor"/>
            <sbf:Behavior>"Low_Speed"</sbf:Behavior>
            <sbf:Flow rdf:resource="sbf#Oscillatory-translational-motion"/>
        </mug:Link>
    </mug:LinkHandle>
    ....
</mug:World>
```

(c) DAML encoding of Function-Flow for the motor/lens housing link for the scanner.

Figure 3: Illustration of MUG's ontological models and how they are used to describe device semantics.

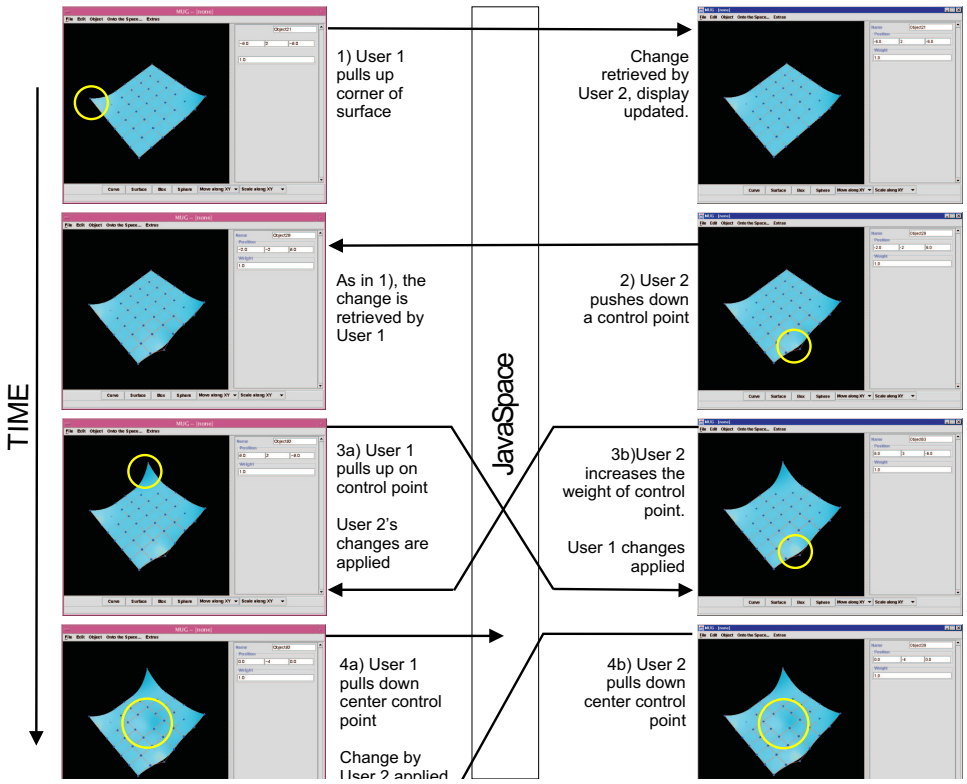


Figure 4: Multi-user manipulation of a NURBS surface within MUG. The user on the left, working on a UNIX system; the user on the right working on a Windows NT system. The changes are propagated to the clients through a JavaSpace while clients continue to work independently.

Each agent is associated with a separate design and is responsible for managing messages describing transformations to its particular design.

MUG Client. The **MUG Client** features two threads of control, as well as a number of internal components. The first thread of control is a GUI thread, it provides user with control as well as the communication to the server. The second thread, serves as the listener to the transform-messages that come from the server. The main set of components that make up **MUG Client** are the **UndoManager**, the **BehaviorManager**, and the **PluginLoader**.

The **UndoManager** keeps track of all changes, whether generated locally or from elsewhere in the collaboration space. The **UndoManager** stores changes as a history tree, allowing the users to walk down one branch of the tree, then backtrack and walk down the other branches, the traversing the entire space searched during the design process. This can help facilitate documentation of design mistakes for future reference.

The **BehaviorManager** binds the user actions (e.g., mouse events, keystrokes) to the software that generates proper responses. The **PluginLoader** allows MUG to hook in additional components written using MUG's internal API. The MUG NURBS library is an example of one of such component.

Figure 5 shows the major components in the MUG software, as well as the flow of information and messages among the server and the multiple MUG Clients.

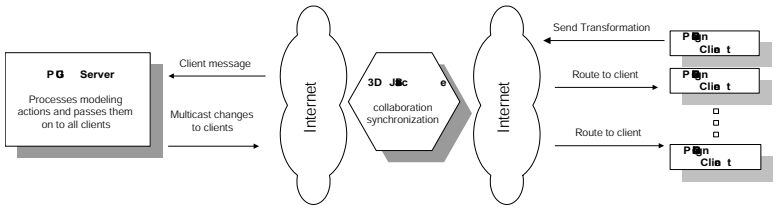


Figure 5: Elements of MUG's systems architecture and the communications among it's different components.

5 Using MUG: Potential Scenario

Figure 6 provides an illustration of how a team of engineers can use MUG to collaboratively describe the semantics of the one DOF scanner. While this example shows the flow of changes between users in a two-person team, MUG can accommodate (theoretically) an *n* person team. In the sequence shown in the Figure, the team imports an unattributed shape model, such as might have been created in a commercial CAD system. They proceed to annotate its function and flow properties, reaching the complete description of the design semantics shown in Figure 7.

6 System Implementation

MUG currently runs on Pentium II/III/4-based machines with Microsoft Windows NT 4.0 or Windows 2000 and Sun Ultra workstations with Solaris 2.6/7/8 that are equipped with Java Runtime Environment version 1.2 and the VRML97, Java3D, Jini/JavaSpaces, Java Media Framework, Java Help, and XML 1.0 extensions. VRML97, Java3D, Java Help, and XML 1.0 extensions.

MUG designs are are saved in DAML format (`.daml`), structuring the information about each component, both geometric and semantic, of the model within the file. In addition to providing a set of Constructive Solid Geometry (CSG) design primitives, MUG can also import pre-existing geometries defined in VRML format (`.wrl`)—such as those created by other CAD systems.

Java3D. MUG's 3D spaces are created using Java3D and its X3D and VRML97 extensions for manipulation of 3D worlds. Our choice of Java and Java3D as the basis for MUG (instead of C++ and OpenGL) was motivated by its ease of integration with other supporting packages for shared worlds and multimedia, as well as by its cross platform abilities. MUG can also make use Java3D Stereo Mode, thus allowing users to interact through a workstation-scale 3D environment.

JavaSpaces. As noted in Section 4, each instance of MUG in a collaborative design session is a **MUG Client**. Each MUG Client performs operations that map entries or templates onto JavaSpaces services. JavaSpaces is the Java based implementation of a "Tuple-Space" paradigm [2]. With MUG, we have created a synchronized 3D tuple-space in which teams of designers can work.

A single **MUG Client** can interact with as many spaces as it needs to. Identities are accessed from the security subsystem and passed as parameters to method invocations. Notifications go to event catchers, which may be clients themselves or proxies for a client (such as a "store-and-forward" mailbox).

The MUG tuple-space features a system for storage of serialized objects, that can be retrieved by any client connected to the design session. Objects stored in the space can be read or deleted, and modification is done by first removing an object, modifying it locally, and placing it back into the space. JavaSpaces handles all of the data transfer among the MUG Clients and the MUG Server—transferring data as native Java objects that describe the 3D design scene and its semantics.

Two categories of inter-agent messages are carried via MUG's JavaSpace: short-life messages and long-life messages. Short-life messages are the ones passed from client to server or agent or vice-versa (e.g., attach to a design session, request a copy of the 3D design, etc.). Long-life messages represent changes to a particular design and used a subset of the Knowledge Query and Manipulation Language's (KQML) performatives (e.g., delete an object, transform an object, change an object property, etc.) and exchange their message content in DAML.

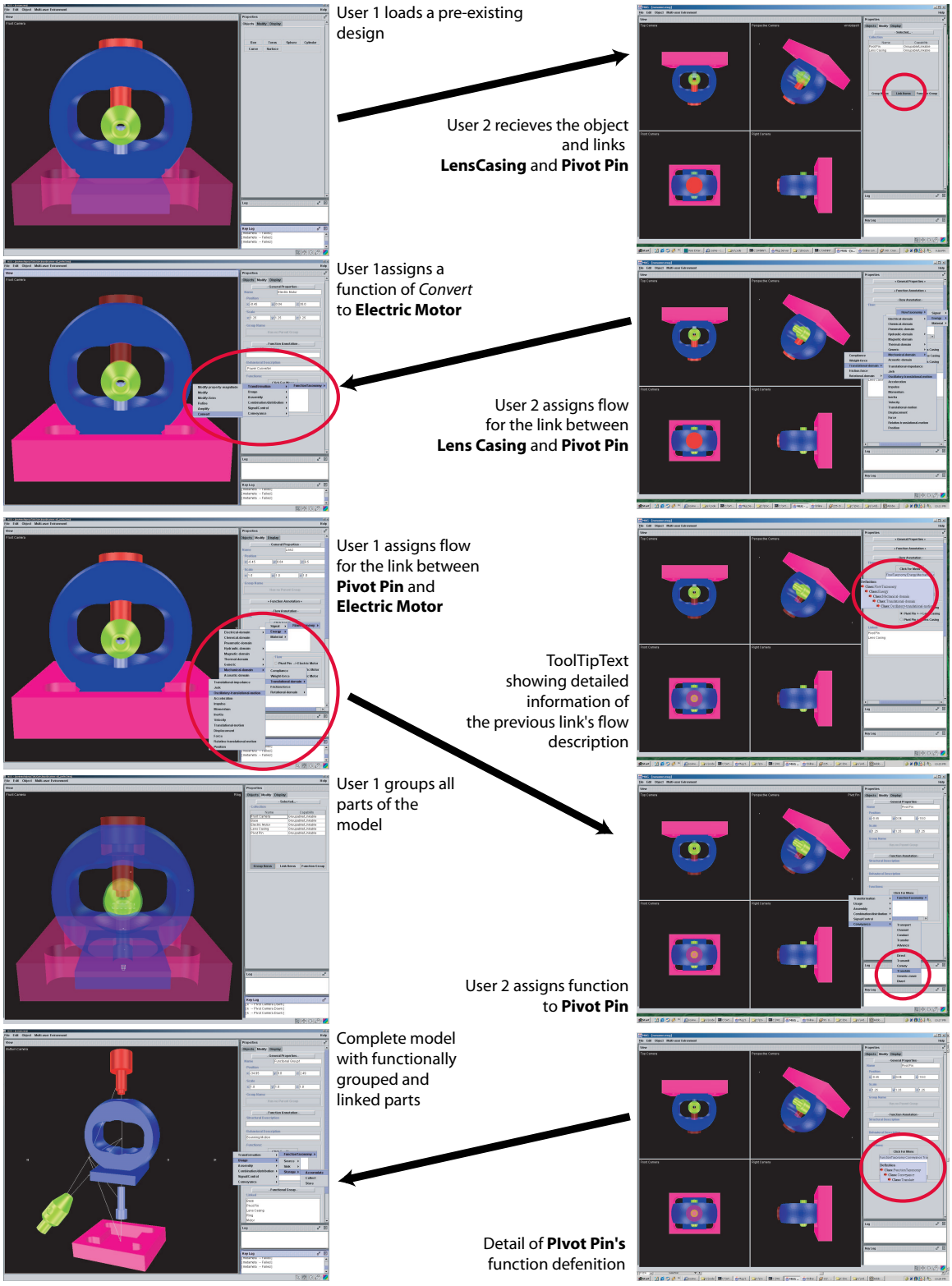
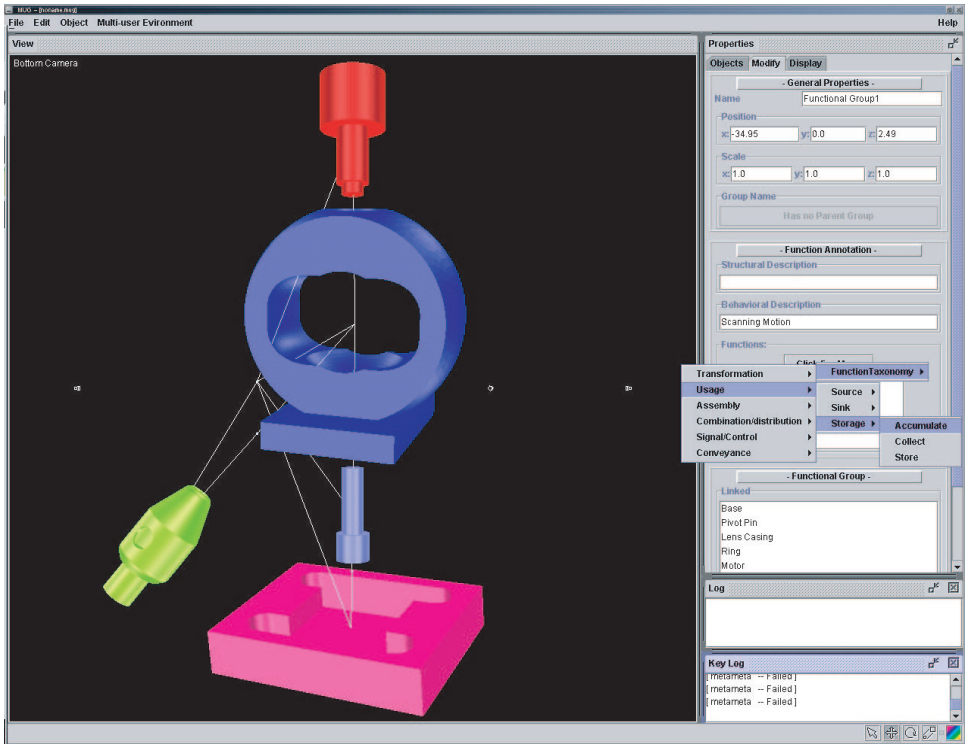
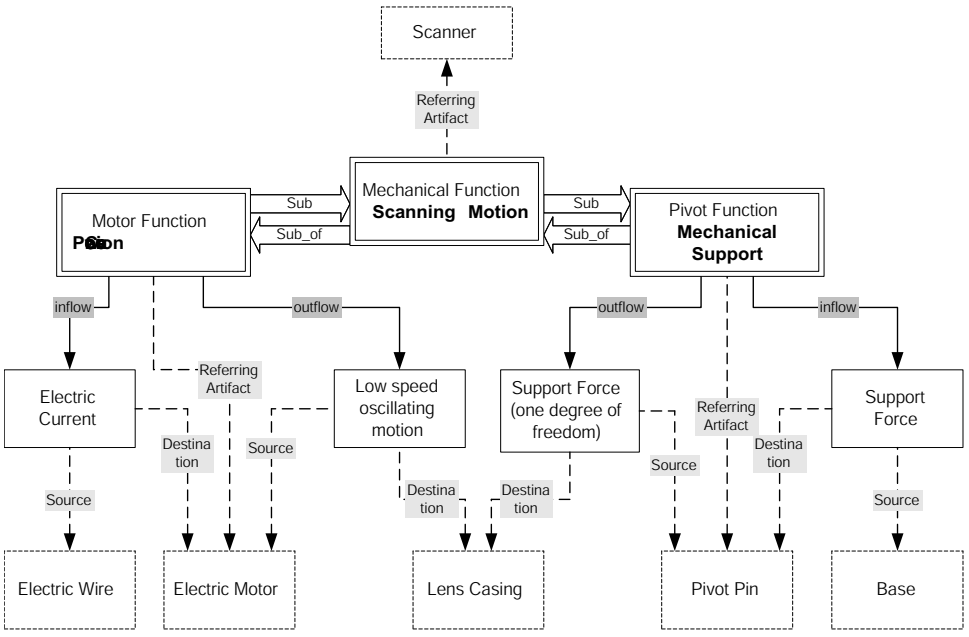


Figure 6: Snapshot of a multi-user session in MUG where a pair of designers describe the semantics of the scanner.



(a) Final exploded view with functional links.



(b) The complete Function-Flow model for the scanner.

Figure 7: Final description of the semantics of the scanner assembly.

Multi-Media Collaboration with Java Media Framework (JMF). MUG uses the Java Media Framework (JMF) to implement a voice-over-IP conference service to carry designers' discussions. In contrast to nearly all commercial audio conferencing tools (peer-to-peer and pairwise), our implementation is many-to-many, i.e., it creates a "conference call" for the *n* members of the design team.

Communications between MUG Clients can be multicast or unicast depending upon available hardware on the host platforms—the JMF's hardware abstraction has allowed us to "write once" and run on Sun and Windows operating systems. The audio conferencing associated with a design are also recorded using the Moving Picture Expert Groups (MPEG) Layer 3 (mp3) ISO/IEC standard format. This allows developers further down the product lifecycle to actually hear design discussions and decisions made by other earlier developers.

7 Discussion

7.1 Using Java for Multi-modal Collaborative 3D

Both our graphics performance and development experiences with Java3D have been quite positive. In our informal experiments, MUG and its Java3D engine running on a Sun Ultra 60 (with Elite m6 graphics) and Dell Workstation 410 (Diamond FireGL2) can easily handle VRML models of complex CAD objects interactively. Figure 8 shows manipulation of a 3Meg VRML model file of a automotive suspension assembly in MUG.

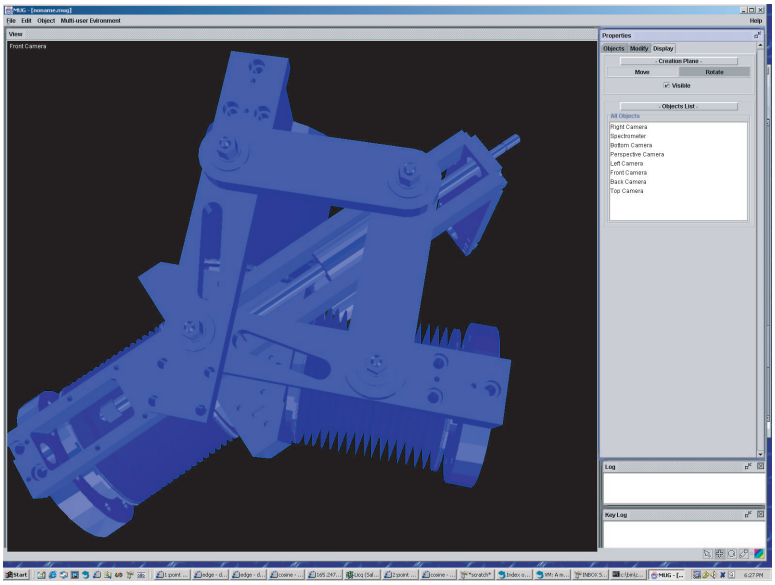


Figure 8: A complex design model of a variable radius spectrometer (a 3Meg VRML file imported from Pro/ENGINEER) being manipulated in MUG.

Our development experience with the JMF has been positive as well. JMF can enable developers to rapidly create multimedia applications, such as shared audio and video. The major issue with the JMF-based audio conferencing features of MUG are ones of scalability. *n*-ary audio conferencing required creation of shared audio tuple-space, where the audio channels from each of the MUG Clients are multiplexed at the audio conferencing server and then broadcast out as a single stream to each of the MUG Clients. This is both computationally intensive and bandwidth intensive. Our informal results so far indicate that, on a 100Mbt LAN, discussions among 3-5 designers are feasible. Latencies make Internet-based audio conferencing very difficult. Our future plans call for experiments with a local gigabit LAN and with collaborating designers distributed over Internet2.

7.2 Analysis of Communication/Computation Tradeoffs

To enable collaborative manipulation of complex mathematical objects like NURBS, we examined the tradeoffs between network bandwidth utilization and computation. The first approach saves computer cycles on the **MUG Client** at the expense of client memory and network bandwidth utilization. In this case, **MUG Clients** operate in a peer-to-peer fashion, each with equal status in the group, propagating the full NURBS mesh to other clients after each change. One issue is that each client has to perform some level of version control on the NURBS object.

The second approach saves memory and bandwidth, but incurs computational costs since NURBS objects must be recalculated at each **MUG Client** as changes are propagated. The major parameters for sharing NURBS surfaces are:

- $|P_u|$ and $|P_v|$: the number of control points for the two defining curves, U and V;
- g_u and g_v, the granularity of the mesh in the u and v directions, respectively.

To estimate requirements for the first approach, if the mesh granularities are is g_u and g_v, the total number of nodes that define the surface is $g_u g_v$. Each node is described by a 24 byte array, that makes up 3 double precision numbers (the $x-y-z$ coordinates of the mesh point). The number of triangular facets that make up the surface is $4(g_u-1)(g_v-1)$. Each triangle is defined by 3 indices which are 4-byte integers each, and 1 normal vector which is a 3 element array of 8-bytes per element: a total of $O(168 g_u g_v)$ bytes of data. When adding the control points: $O(32|P_u||P_v|)$, where 32 is an 8 bytes describing weight and 24 bytes describing position, bringing the total memory storage and transmission requirements for the first approach is $O(168 g_u g_v + 32|P_u||P_v|)$—or $O(m^2)$, where $m = \max(g_u, g_v, |P_u|, |P_v|)$. For the second approach, the upper time bound on the amortized computational costs are $O(n g_u g_v)$ where n is the total number of distinct changes done to the surface.

We ran some empirical tests on a 7x7 NURBS patch with $g_u = g_v = 50$ and calculated that, when using the first approach, 188KB of data were transferred to each **MUG Client** every time a change was made. Using the second approach, our Intel-based workstations recomputed the whole surface nearly interactively and MUG exchanged only 100 bytes of data among clients. In most cases, surface changes are minor and only a small, localized patch of the NURBS needs to be recomputed.

Based on these brief studies, and in the interest of minimizing overall bandwidth utilization (and save room for the requirements of a high-fidelity audio conversation), MUG uses the second approach: instead of storing complete objects, we store the forward and backward changes that are applied to the objects. These changes are propagated through the network when we work in a multi-user mode. The MUG NURBS library, for example, sends updates whenever a control point is moved, i.e., information describing this only this control point, its object-id, its previous position and its new position are stored; the NURBS mesh is then recalculated on each machine individually.

7.3 The Potential for DAML

We believe the real contribution of MUG is to illustrate one methods of uniting interactive 3D modeling with the creation of shared semantics. The DAML and Semantic Web efforts are bringing practical knowledge representation solutions to real-world problems. We see the use of DAML in MUG as a first step toward building tools to let designers easily model with deep semantic detail and high graphic fidelity. DAML is intended to be flexible, adaptable and enable fluid growth of its ontologies—hence, readers are encouraged to take, refine and extend the ontologies we have created for MUG for use in other applications that need electro-mechanical or AEC semantics.

8 Conclusions, Contributions and Future Work

This paper presented MUG, our approach to collaborative authoring of design semantics within multi-user 3D environment. This work begins to unite ideas from traditional engineering design with recent work from the Semantic Web and knowledge engineering community. With MUG, engineers have a new way of collaboratively capturing the design intent inherent in the design process. MUG lets users author the structural, behavioral and functional knowledge about a design in a 3D virtual environment. Ontologies used for defining design semantics, as well as the result of the design activity, is stored in DAML, providing a consistent and semantically well-grounded format for interpretation by both computers and human beings.

Our near-term work to improve to the existing MUG environment will include facilities for authoring and manipulating design ontologies on-the-fly; the refinement of our DAML-based MUG file representation to encompass arbitrary design

domain ontologies as well as VRML97 and ISO STEP-based descriptions of geometric objects. We also plan to perform a series of user studies, both with student users and practicing engineers, to determine how MUG and tools like it can be best used to improve the design process.

Longer term, we believe that our work indicates several possibilities for new CAD-based applications. The first is to migrate geometry-centric CAD tools toward include the authoring of more formal descriptions of design knowledge. A second area to explore is the integration of CAD activity with CSCW systems that facilitate group activity. Our belief is that CSCW needs to be intelligently integrated into the activities of engineers, enhancing CAD by enabling the collaboration tools to be aware of the domain-specific design context in which the collaboration occurs. Our future work will explore both of these questions and, hopefully, continue to reveal more to us about the true nature of collaborative design.

9 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/IRIS-9733545 and ONR Grant N00014-01-1-0618. 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] L. Al-Hakim, A. Kusiak, and J. Mathew. A graph-theoretic approach to conceptual design with functional perspectives. *International Journal of Computer Aided Design*, 32(14):867–875, December 2000. Special issue on Conceptual Design.

[2] Nicholas Carriero and David Gelernter. *How To Write Parallel Programs, A First Course*. The MIT Press, 2nd edition, 1991.

[3] L. Eggli, G. Elber, and B. Bruderlin. Sketching as a solid modeling tool. In Jaroslaw Rossignac, Joshua Turner, and George Allen, editors, *Third Symposium on Solid Modeling and Applications*, New York, NY, USA, May 17-19 1995. ACM SIGGRAPH and the IEEE Computer Society, ACM Press. Salt Lake City, Utah.

[4] L. Eggli, C. Y. Hsu, B. D. Bruderlin, and G. Elber. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design*, 29(2):101–112, February 1997.

[5] Marti A. Hearst, Mark D. Gross, James A. Landay, and Thomas F. Stahovich. Sketching intelligent systems. *IEEE Intelligent Systems*, 13(3):10–19, May-June 1998.

[6] T. Hwang and D. Ullman. The design capture system: Capturing back-of-the-envelope sketches. *Journal of Engineering Design*, 1(4):339–353, 1990.

[7] H. Lipson and M. Shpitalni. Conceptual design and analysis by sketching. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, 14(5):391–402, November 2000.

[8] C. J. Moore, J. C. Miles, and D. W. G. Rees. Decision support for conceptual bridge design. *Artificial Intelligence in Engineering*, 11(3):259–272, July 1997.

[9] S.F. Qin, D.K. Wright, and I.N. Jordanov. From on-line sketching to 2d and 3d geometry: a system based on fuzzy knowledge. *International Journal of Computer Aided Design*, 32(14):851–866, December 2000. Special issue on Conceptual Design.

[10] David Serrano and David Gossard. Tools and techniques for conceptual design. In Christopher Tong and Duvvuru (Ram) Sriram, editors, *Artificial Intelligence in Engineering Design: Design Representation and Models of Routine Design*, volume I, chapter 3, pages 71–116. Academic Press, 1250 Sixth Ave, San Diego, CA, 1992. ISBN 0-12-660561-0.

[11] Malgorzata Sturgill, Elaine Cohen, and Richard F. Riesenfeld. Feature-based 3-d sketching for early stage design. In A. A. Busnaina, editor, *ASME Computers in Engineering Conference*, pages 545–552, New York, NY 10017, September 17-20, Boston, MA 1995. ASME.

[12] S. Szykman, J. W. Racz, and R. D. Sriram. The representation of function in computer-based design. In *ASME Design Engineering Technical Conferences, 11th International Conference on Design Theory and Methodology*, New York, NY, USA, September 12-16, Las Vegas, NV 1999. ASME, ASME Press. DETC99/DTM-8742.

[13] S. Szykman, J. Senfaute, and R. D. Sriram. Using xml to describe function and taxonomies in computer-based design. In *ASME Design Engineering Technical Conferences, 19th Computers and Information in Engineering Conference*, New York, NY, USA, September 12-16, Las Vegas, NV 1999. ASME, ASME Press. DETC99/CIE-9025.

[14] Simon Szykman, Ram D. Sriram, and William 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):3–11, March 2001.