Conceptual Design for Mechatronic Assemblies

Santiago V. Lombeyda* William C. Regli†

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

This paper presents an approach to support computer-aided conceptual design of mechatronic assemblies in a collaborative, multi-user environment. We describe a system, Conceptual Understanding and Prototyping (CUP), that allows a team of design engineers, collaborating over the Internet, to develop a high-level structure-function-behavior (S-B-F) description of an assembly in a VRML-based virtual environment. Our goals is to enable users to navigate intricate product data management (PDM) and case-based design knowledge-bases; providing the ability to perform design at conceptual level and have intelligent CAD tools that can draw on details from large repositories of previously archived designs.

This work furthers research efforts in computer support for collaborative design activities—in particular drawing on work from in Human-Computer Interaction (HCI) and Computer Supported Collaborative Work (CSCW). In addition, we envision CUP and tools like it to be the network interfaces to next-generation engineering PDM systems and CAD data-bases. We are deploying CUP as query interface to the National Design Repository (`http://repos.mcs.drexel.edu`). This will enable CAD users to interrogate large quantities of legacy data (models and assemblies) and identify artifacts with structural and functional similarities—allowing designers to better perform case-based and variant design.

Keywords: Conceptual design techniques, Assembly modeling, Collaborative/distributed design, Virtual Environments and Prototypes, Computer-Aided Design, Conceptual Design, World Wide Web, Network-Enabled CAD/CAE.

1 Introduction

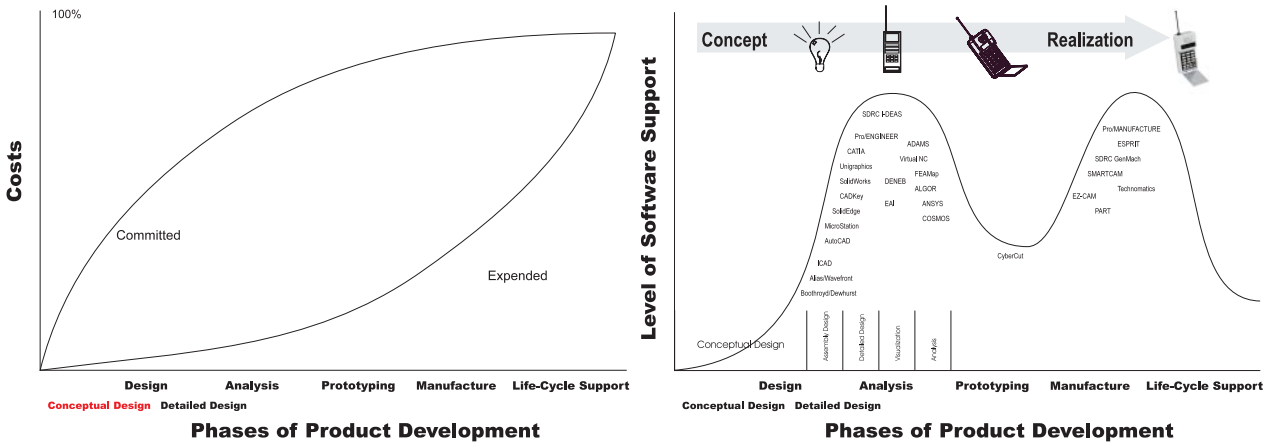
It is conservatively estimated that more than 75% of engineering design activity is case-based design [27]. An engineer or team of engineers, when posed with a new problem, synthesize solutions through a combination of analogical reasoning, experience, and search through design information archived in the corporate files and described in product and component catalogs. Over the past decade, 3D Solid Modeling has become a critical element of the product realization process—leading to successive generations of increasingly robust software tools for *detailed* design and engineering analysis.

The goal of this research is to develop novel techniques and tools to support *Conceptual Design*. Conceptual Design is the initial phase of product development, when product development teams (consisting of design engineers, manufacturing engineers, marketing and management personnel) determine the essence of a new product. As has been often documented, and shown in Figure 1 (a), the design phase of the product realization process has a tremendous impact on the life-cycle cost of a product. However, as illustrated in Figure 1 (b), there are few tools to support 3D conceptual design.

*Present Address: Center for Advanced Computing Research, Mailstop 158-79, 1200 E. California Blvd. California Institute of Technology, Pasadena CA 91125 USA. Email: `slombey@caltech.edu`

†URL: `http://www.mcs.drexel.edu/~regli`; Email: `regli@drexel.edu`.

During Conceptual Design, teams of designers may begin to develop a new product by sketching its general shape on paper. This "back of the envelope" approach is key aspect of the creative thought process—once completed, one has a clearer idea of what one is creating and can proceed to drafting or CAD activity. Previous research in areas such as conceptual design of graphical user interfaces (GUIs) [13, 18] and cased-based reasoning and case-based design [26, 25, 20, 10, 1] are relevant to our efforts. For example, much of the work in case-based reasoning has focused on developing symbolic representations for engineering knowledge about function and behavior, as well as algorithms for indexing, retrieving, and adapting design cases stored with these representations.



(a) (b)

Figure 1: Decisions made at the conceptual design phase have great impact on product viability and cost (a). However, there are few software support tools for conceptual design (b). Figure (b) illustrates level of software support for various aspects of the product realization process (some commercial tools are listed).

This paper presents **CUP**, a Web-based, 3D modeling tool for **Conceptual Understanding and Prototyping**. CUP allows teams of users, during the conceptual phase of design, specify a spatial layout of components and sub-assemblies; it allows users to specify structural, functional, behavioral information about components and sub-assemblies; and it provides mechanisms for capturing textual information about the design's intent and designers' preferences.

The paper is organized as follows: Section 2 reviews related work for this project, emphasizing recent results in Human-Computer Interaction and Computer-Supported Collaborative Work that are relevant to engineering design. Section 3 details our approach to building the CUP tool, our design methodology, and our description of our design goals. Section 4 describes an example of how CUP may be used to create a conceptual sketch for a design problem. Lastly, Section 5 summarizes our conclusions and gives some of our plans for future research.

2 Related work

2.1 Collaborative Design

Gruding and Poltrock [12] note that the expanding scope of Computer-Supported Collaborative Work (CSCW) has grown to include multi-disciplinary efforts in a diverse number of fields. Much of the current existing work on collaborative computer-aided design involves the use of immersive virtual reality systems. This work includes efforts at Stanford [7, 16] on collaborative production modeling and planning using the Responsive Workbench; and work at University of Illinois at Urbana Champaign on vehicle design using a CAVE. In these environments the user is able to directly explore and maneuver around the objects in the environment, as well as to relate

with other participants with a tactile virtual reality space. These immersive systems require high band-width networks and considerable amount of custom graphics hardware [4, 17].

Research on object selection has included the ability of a user to select an object while in an immerse environment. In Bowman [2] the authors describe a combination of techniques such as the 'arm-extension' and 'ray-casting' paradigm with which the user is able to select and manipulate objects. Research on collaborative object manipulation has also been studied by Pang and Wittenbrink in a system called Cspray [22]. The authors created a protocol by which users may request/relinquish control of objects, users may share exchange views, and may modify or may lock objects. The focus as well on related issues such as network latency and three dimension pointers in a collaborative space.

Other research issues have included how to handle multiple pointers in an environment to manipulate objects in a more intuitive and efficient manner. Zeleznik and Fosberg [28] showed that users using multiple simultaneous input perform better than with a single input mode. This was also taken in consideration as the prototype was developed, however due to environment restrictions different approaches were take, though they rely on much of the same intuition as multiple pointer manipulation.

2.2 Research in Human-Computer Interaction (HCI) and Graphical User Interfaces (GUI)

An integrated CAD environment, ideally, should increase increase drawing speed and accuracy over traditional manual drawing/drafting methods and enable a wide variety of digital analysis. Mitta and Flores [21] explore how different interface designs and input methods have repercussions in CAD productivity. The results of their study are conclusive towards one specific interface design. However strong points are made in favor in terms of more intuitive and customizable interfaces. They also propose a method to study CAD productivity and efficiency in terms of test studies doing specific tasks. A similar point is also maintained in [8], who argues that CAD users are more productive when they have the ability to customize the interface for specific needs. Other related HCI work includes research on different methods to display information (such as three dimensional data trees), handling information transmission delays through a network or web [14]. There are several research efforts which have created complex virtual reality collaborative systems across ATM and Intranets [3, 4, 19, 22].

2.3 Variational and Case-based Design

Case-based and Knowledge-Based Systems for engineering and design have been an active research area for the past 15 years [20]. CADET [26, 25] and its descendent projects focused on conceptual design and solved problems using relationships that capture function, structure and behavior. CADET builds solutions to new design problems from pieces taken from previous design cases. CADET's design models were behavioral and functional representations, with input to the system consisting of symbolic descriptions of the desired device along with some physical constraints. The designs knowledge-base of CADET is a store of function, behavior, and the device's structural relationships. Indexing and retrieval is performed using linguistic descriptions of these properties as well as queries on the symbolic information and its parameters. The retrieval and indexing methods are based on variations of graph matching and support retrieval at varying degrees of abstraction.

Goel et al.'s KRITIK and its descendent systems [10, 1, 9, 11] operate on design problems using a case-base of designs represented by symbolic component descriptions, their relationships and behaviors. A central contribution of KRITIK was the formalization of a *structure-behavior-function model* (S-B-F) for designs, where design cases can be indexed according to the functions they deliver. The functional representation is hierarchical, consisting of a component-substance model to capture of structure and performance of a given device. KRITIK's design domain is not linked to specific CAD or solid model geometry and topology specifications (such as are captured in current CAD and PDM systems) and is limited to devices whose functions can be characterized as a flow of substances between components. The more recent work as extended many of the earlier KRITIK concepts, however their powerful reasoning techniques are still primarily symbolic and have not been coupled with detailed engineering data.

In a survey of recent work on variational and case-based design, Fowler [6] makes several observations: better abstract models are needed for mechanical artifacts so that function information can be stored in the CAD knowledge-base (in much the same way that functional indices are computed in KRITIK). Complex issues need to be considered to develop systems for automatically retrieving and applying existing designs to solve new design

problems. CAD systems should soon be augmented with CBR/CBD techniques and, if done properly, will lead to great benefits to designers.

3 Approach

Our goal is to develop software tools to support the conceptual phase of design. Our approach draws on current Software Engineering practice for developing interactive software environments. Supporting Conceptual Design for mechanical assembly means providing the ability to model assemblies without having to define, in detail, every component or feature. To satisfy these requirements, CUP must poses the following characteristics:

1. **CAD-based:** Users require basic CAD capabilities for creation of designs and object primitives, with the added capability of embedding structural, behavioral, and functional characteristics.

2. **VR environment:** There are many advantages to using a Virtual Reality environment where the user has the capability of exploring a design as if working at a conceptual workbench [16]. Furthermore, the use of a virtual system adapts itself very well as a paradigm for collaborative interaction and work [17].

3. **Communication:** In order to create a multi-user environment, as well as be able to access online design knowledge-bases, the conceptual design environment must have the ability to access network communication protocols.

4. **Portability:** The environment ideally would give the designer the ability to work on any platform that has internet access and a browser. For our purposes and goals it is assumed that networking capabilities are available. Furthermore we envision that this tool will act as a client of server which keeps control and design managers across the network. Thus some tasks will be performed by the tool itself, but some more external conditions must be determined by an outside omniscient server, and thus out of the scope of this study. It is also assumed that there exists a direct access to a design knowledge-base to respond to queries based on structural, functional, or behavioral properties. The server must be able to handle this kind of information, do any format filtering, and respond to the client in protocols to be introduced.

Specifically, we present a four-step design process to achieve these goals:

1. **Requirement Specifications,** including a description of the interface specifications in terms of goals and requirements. Our specifications may be divided between *functional requirements* and *non-functional requirements*. Functional requirements encompass all prerequisites directly related with the ultimate functionality and purpose of the tool to be designed. Non-functional requirements do not affecting the pure function of the object, but affecting other areas of the design (for example specific software and hardware platforms under which the model is to run).

2. **Interface Modeling and Design** involves an in-depth analysis of how to approach the different issues raised by the Requirement Specifications. Initially, we proposed a model consisting of a description in terms of input and output mechanisms, internal data structure and flow, as well as functionality and behavior. We then developed evaluation criteria for the domain and test the initial model against these criteria—refining the model as needed.

3. **Implementation** of CUP will be based common Internet technologies: Virtual Reality Modeling Language (VRML), Sun Microsystems' Java language and Netscape's JavaScript languages. The intention is to create a downloadable tool that can be executed inside a VRML-enabled web browser.[1]

4. **Testing** CUP involved two phases. First, via interaction with human users, we evaluate the functionality and usability of the tool. Second, we verify that the product itself is consistent and works as expected. During testing it is important to keep in mind the target group and, as outlined in [8], that though there are different level of users—some fields do present more prior knowledge in the average user than others. For instance users which are trying out software development environments are more likely to take advantage

[1] Internet browsers such as Netscape Navigator 3.0 and higher may make use plug-ins such as Cosmo Player by Silicon Graphics Incorporated or Sony VRPlayer.

of shortcuts, scripting, and the ability to create macros, versus office assistants whose sole purpose is to use a word processor. As users test the interface they may evaluated for efficiency in different aspects of productivity, as well as to gather feedback and what is well received and functional, versus that what is misunderstood or useless.

3.1 Requirements

The desired **functional requirements** of the conceptual design environment include:

- The user must be able to travel the work environment, and thus gather integral information about the objects present;
- The user must be able to perform some level of conceptual design. From this it follow that the tool:
  - Offer a 3-dimensional space supporting basic design;
  - Ability to group combine objects to create a more complex entity;
  - The capability of specifying specific properties to a particular object, in textual or other form;
  - Ability to link objects together creating a hierarchy based on physical, functional, or behavioral characteristics;
  - Allow users to specify or choose a specific object, collection of objects, or group, and modify its properties;
- The tool must support the vision of collaborative work by several users simultaneously. This in itself means that this model must include such characteristics as:
  - Ability to display the presence of different actors within the same design environment;
  - Be able to allow or prohibit access to specific objects given private ownership or current control by a different user. Notice however that particular ownership/control of on object is given by a control manager outside this tool itself (server).

For the tool itself, our goal is for it to be widely accessible (platform independent) and Internet-based. **Non-functional requirements** for the interface include:

- Should allow the user to view information on specific objects, groups, or users;
- User should be allowed to choose and change specific properties of the object such as color, transparency, or other forms of rendering mechanisms (i.e. wireframe view);
- The interface must have a mechanism to display change performed by other users, as well as query of specific environment properties may/may not want to be affected (i.e. a user chooses to have a wireframe view for one specific object. If in the context of a discussion all users may wish to see exactly what she/he is seeing.);
- Should allow the user of a whiteboard where to perform work outside the view of other users. Work may then be ported to the shared world Different views (cameras) should be offered to the user, including the ability to see though the "eyes" of a different user.

3.2 Interface Design and Implementation

We describe CUP and detail information on the input and output control interfaces, internal structure, and the functionality and behavior properties of the conceptual design tool.

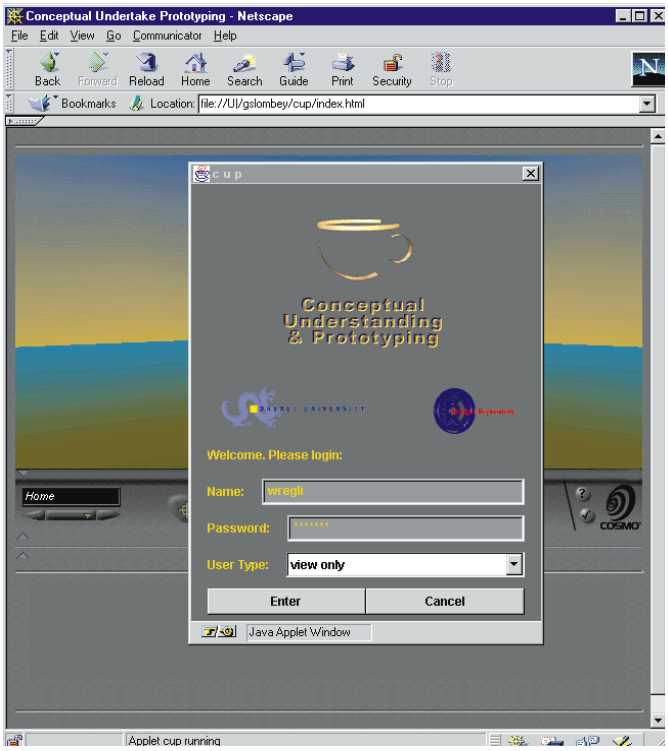


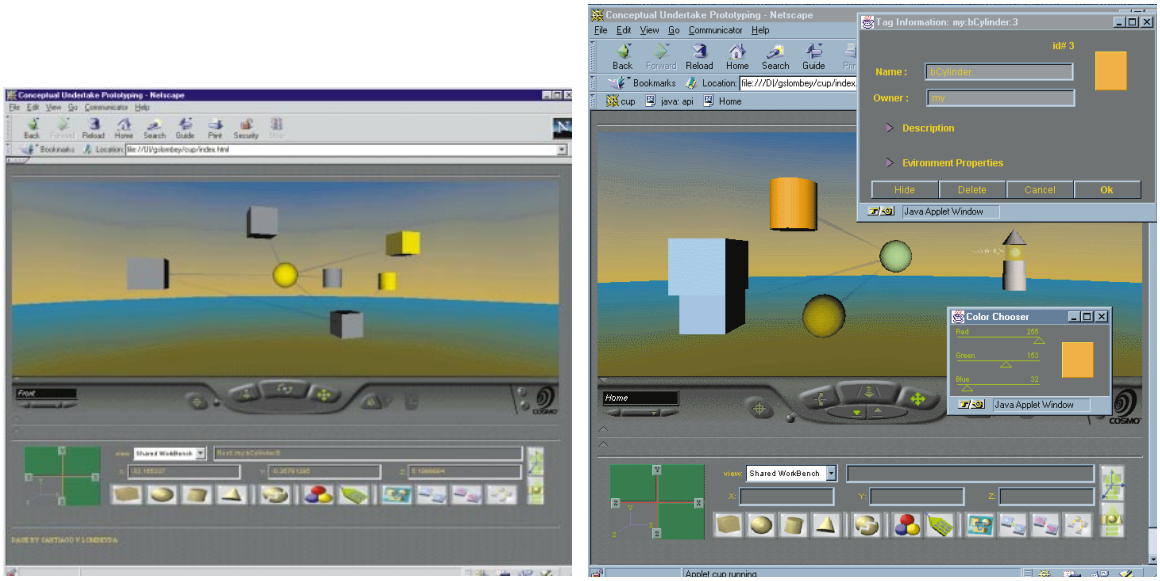
Figure 2: The initial entry point into the Collaborative and Conceptual Design Environment (CUP) consists of a login screen and a description of the user mode.

Privileges and Security. Four different privilege levels of user may be established: *administrators, privileged users, users*, and *guests*. There are three different view modes that control the ability of users to access different views, tools, etc.: *collaborative mode*, *search mode*, and *view only* mode. Collaborative mode enables users full use of the capabilities of the interface. The second mode, search mode, gives the user the ability to use a 3D whiteboard to create a 3D conceptual model and use it as a query to an engineering knowledge-base. The final mode is that of passive viewer, which only requires guest status, and that permits the users to travel the work space without any design privileges. A view of the initial CUP startup and user login screen is shown in Figure 2.

In collaborative mode users will enter a shared workspace and be able to fully participate with other users in the collaborative design. Users will also be able to access their private workspace, and search the repository from the private workspace, or the shared world. Notice that while search is a tool/option offered which requires certain privileges, we also have the tree view which always appears as a help view for those modes in which design is allowed.

Objects. Our view is that a conceptual design environment consists of a very elemental CAD system in which the user can design without focus on details and yet be able to introduce enough information so that a full design may be evolve from this work . For this purpose we propose an environment in which the user is able to create and manipulate primitive three dimensional primitives (blocks, cylinders, etc.), and explain further characteristics of these objects textually, as well as correlate relationships between these objects. This will be accomplished by offering to the end user a limited CAD environment with the abilities to tag, link, and group elements.

Tags and Links. All objects in the design space will have a one-to-one relationship with a tag containing information on its structure, functionality and behavior as well as textual descriptions for the component. Objects may be linked to each other (uni-directional or bi-directional links) establishing structural, functional, or behavioral links among components. An example of the linking and tagging facility is show in Figure 3. Finally,



(a) (b)

Figure 3: Screen shots of CUP and some of the facilities for linking and tagging objects.

a selection of multiple objects may be grouped to create a conglomerate which does not only hold the specific characteristics of each of its components, but a tag of its own, with its corresponding set of information.

Multi-User Collaboration. The second challenge is to introduce a collaborative environment to this process. We propose that this be done by allowing several users to participate in common design processes, with protocols similar to those studied previously [22]. Users will have the ability to participate in a shared workspace, as well as make use of a personal work bench or white board.

When participating in a shared workspace it is assumed that a parallel system of communication is also being used (text chat, web phone, etc.), but at the same time users interact in the design space, viewing the design at work, as well as being able to see the others users present in the world through their avatars. As users interact, they are able to created, modify, and even destroy objects within the design, following a protocol where objects may be requested, relinquished in control, or even protected or hidden from other users.

A tree view is also offered which allows the user to view relational correspondence between objects such as grouping and links in a three dimensional tree structure. This view works in conjunction with shared and private workspaces. Users will enter this interface by accessing a URL, which will automatically start the CUP interface. The user will be required to login using a name and password, as well as enter the sessions type they will be performing under. The three different session types are: collaborative work, view only, or search repository.

Internal Structure. The internal structure of CUP is three layers: *communications layer*, *internal data management and event handling layer*, and the *display layer*. As show in Figure 4, information may flow across these layers back and forth from the user to the environment control management in the internal client itself and at the server (structure of the server not relevant). The common thread that brings this layers together is a Java applet[2] which is responsible for creating sockets for communication with the server, up-keeping the representation of data, and communicating with the vrml browser. To elaborate:

1. The *communications layer* performs asynchronous communication between agents in the environment and the session server, as well as to the internal representation layer and to the display layer.

[2] An applet is a Java application running within a Web browser

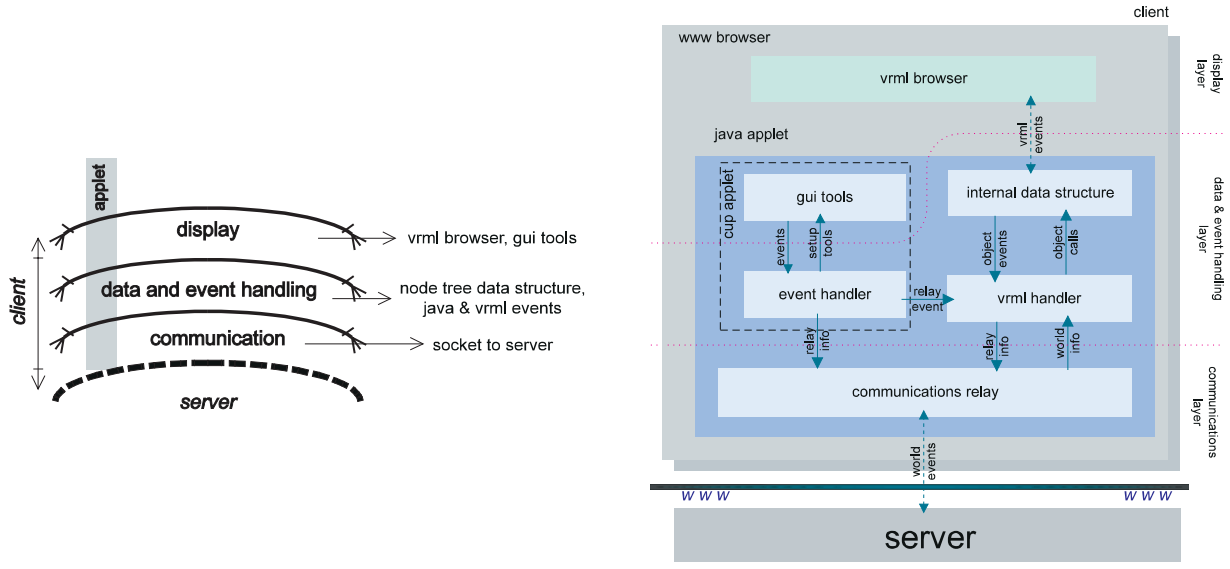


Figure 4: CUP internals.

2. The *internal representation layer* represents a hierarchy of tree which includes environment variables and the design structure. The design structure is a hierarchy of objects which includes such information as properties as well as control and ownership information.

3. The display interface is responsible for managing the environment, and thus event driven. This layer has several states available and dictate what the user is able to do and see at each of them.

Input and Output Interface. The interface is be divided in two areas: the top, a VRML browser capable of displaying a design/world. All object/design output will be displayed through this browser. The second area of the interface consists of the tools applet—responsible for most of the interaction to and from the user. This is implemented by using the External Authoring Interface (EAI) between the VRML browser and Java applets running inside the web browser. The VRML EAI provides a set of Java classes to establish connections between the VRML browser's objects/events and the outside world (in the browser). Through the EAI, Java applets can modify, add, or deleting nodes in the VRML world; or handle changes or interaction among nodes within the VRML world—as shown in Figure 4. For example, the EAI handles creation, manipulation, destruction, tagging, linking, and grouping of objects. The VRML browser is the display mechanism for the world and the event handler for interactive selection and manipulation events.

Data Flow. Data flows across this interface as structures of objects and nodes, pictured in Figure 5. This information flows across the data structure according to actions taken by the user, or outside the interface and relayed by the server. Thus, much of the data flow in terms of design objects is between a class `VrmlHandler` and the browser. This will be trigger by actions caught within the interface, or relayed to the interface from the browser, as well as from the communications Relay class, which gets mapped to the applet as well.

The main data structures being handled represents trees of data structures that closely resemble VRML nodes. In fact these data structures have direct connections with actual objects events. These objects comprise all the information needed for the purpose of conceptual and collaborative design, such as tags with the structural, behavioral, and functional descriptions, as well as information of ownership, control and protection. Links and groups are treated as objects as well, with the different that they have specific characteristics which allow them two point to two (in the case of links) or more objects (groups).

The upper most generic class from which this structure inherits is the `VrmlObject` class. This class is abstracts, which means that no object may be instantiated as being of type VrmlObject, but most be of some

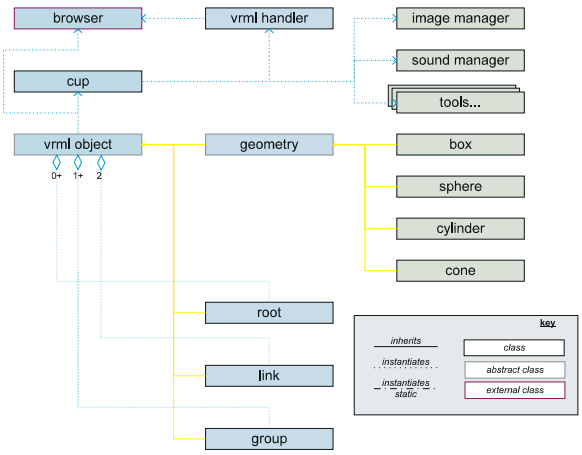


Figure 5: The information classes and methods developed in Java to support CUP.

derived class. A `VrmlObject` has a characteristic a central node, a position accessor method, a tag, and a parent pointer.

There are four main classes which derive from `VrmlObject`: `Geometry, Root, Link, and Group`. `Geometry` class is abstract as well, and is the base class for all shapes to be created and handled in this package. This class has an initializer which is responsible for creating transformation nodes, appearance nodes, and sensor nodes, all of whose change events are store to be used by the `VrmlHandler`.

The `Link` class is similar to that of a `Geometry`, the difference that the parent pointer is always the root, and has two more pointer for `from` and `to` links to `VrmlObjects`. The tag in this case to be used will be of the type `LinkTag`, rather that the generic `Tag` class. Finally the root and group nodes contain vector of object nodes. This hierarchy of classes is described in figure 11.

As objects are created, they are introduced into the world by adding them as children of either the root, or a parent already existent. From these objects, events are captured so as to be able to modify them at a later period, or to be able to capture selection or drag events. It is important to add that these objects are not single nodes, but usually a composition of several embedded nodes such as transform nodes, sensor nodes, group nodes, geometries, appearances, and shapes (VRML nodes). Existence of objects may easily be traced by inspection of the display browser.

Implementation Specifics. CUP runs on Pentium II-based machines with Microsoft Windows NT 4.0 and Sun Ultra workstations with Solaris 2.6. The current implementation is based on Java Version 1.6, Netscape Navigator Version 4.05, and CosmoPlayer Version 2.1. Functionality on the Solaris platform is limited due to the lack of a plug-in compatible VRML browser (we are currently using the VRWave stand-alone browser).

4 Testing and Example

The final interface went through a process of redesign and reevaluation. The testing was done by members of the National Design Repository Project (`http://repos.mcs.drexel.edu`) and Geometric and Intelligent Computing Laboratory (GICL, `http://gicl.mcs.drexel.edu`) at Drexel University. This initial testing was largely informal—done on the basis of interview and feedback from members of the laboratory team using the tool.

CUP is primarily concerned with the creation of conceptual designs of *mechatronic systems*: electro-mechanical systems that combine electronics and information technology to form both functional interaction and spatial integration in components, modules, products, and systems. Typical examples of mechatronic systems include automatic cameras, miniature disk drives, missile seeker heads, and consumer products like CD players, camcorders, and VCRs. These designs include mechanical and electronic components—such as the CAD model

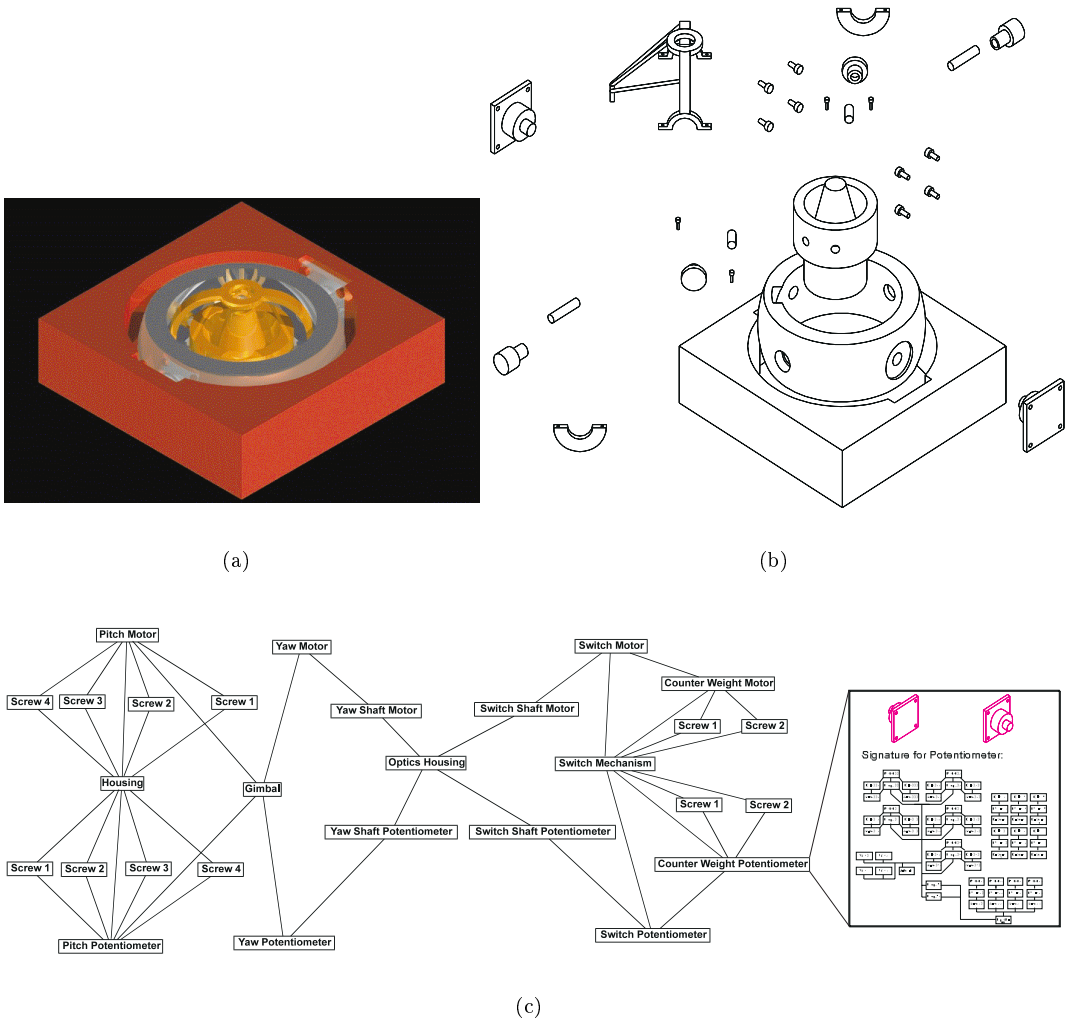
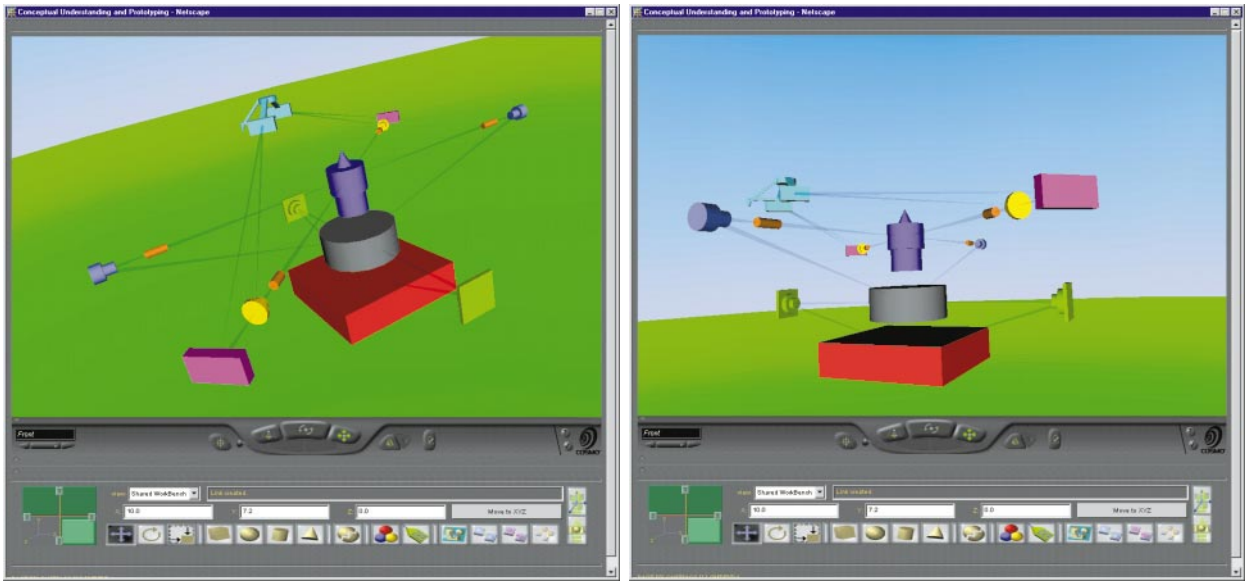


Figure 6: An example of a mechatronic system: a simplified missile seeker assembly along with its assembly structure.

(simplified) of a missile seeker assembly pictured in Figure 6, and related metadata (process and assembly plans, documentation, etc).

This simplified seeker assembly might be one of many dozens stored in a corporate design data/knowledge-base. A design team, faced with the task of creating a new seeker, might want to interrogate the CAD knowledge-base and examine previous design cases that might be relevant to this new problem. Examining this legacy data can prove time consuming and tedious, unless one knows exactly what one is looking for.

As illustrated in Figures 7 (a) and (b), CUP allows a designer to quickly sketch out, in 3D, the major components and structural relationships in the assembly. Rather than performing detailed CAD to create a draft design (detailed CAD modeling for this model took several days), designers can, in a matter of minutes, build a conceptual design. This conceptual design can then be used as a starting point for further refinement or as a query to the design knowledge-base. CUP also, via the attributing, tagging and labeling features, helps designers capture the design intent and to build a structure-function-behavior (S-B-F) model of the artifact.



(a) (b)

Figure 7: Illustrations of a conceptual design for the missile seeker pictured in Figure 6.

5 Conclusions

Research Contributions. This research represents our initial efforts to develop a system for the conceptual design of mechanical assemblies. In doing so, we have created a novel form of modeling system and introduced new object manipulation methods to support the unique user interaction needs in collaborative/conceptual design environments. It is our belief that CUP can been seen as component in a conceptual design environment—enabling users to create a knowledge-level description of the design without having to perform detailed CAD and solid modeling.

Future Directions. We are extending this work in several significant ways. At the time of this writing, these thrusts are each in the development stages:

- **Multi-User Conceptual Design:** Building multi-user collaborative design functionality via CUP's network-based Virtual Reality environment could prove to be a powerful compliment to traditional CAD and CSCW tools. The current CUP does not implement a multi-user environment—adding this functionality is part of our current development efforts.

- **Access to Design Repositories:** Increasingly, engineers depend on Product Data Management (PDM) systems, large-scale engineering digital libraries and knowledge-bases of CAD solid models to perform their job [24, 23]. This involves searching through vast amounts of corporate legacy data and navigating manufacturer's catalogs to retrieve precisely the right components to assemble into a new product. The complexities of this trend are compounded, as team members from different companies collaborate over computer networks on joint engineering projects. Presently, however, design storage and retrieval systems (such as commercial databases and Product Data Management systems) are limited in their ability to capture design intent and reason about CAD knowledge—often relying on the textual annotations of the designer, or the part's filename, for storage and retrieval purposes. In some cases [15], ontology-based classification schemes have been employed. However, these require that all users share a common ontology for representing the data and know, a priori, the precise classification of the item of interest.

It is our eventual goal to integrate CUP with the National Design Repository (`http://repos.mcs.drexel.edu`) and provide facilities for users to add their own ontology information to the repository's knowledge-base.

- **CUP as 3D "Freehand Sketching":** Other researchers have approached conceptual design as a freehand sketching recognition problem [5]. In this work, we have developed a 3D modeling approach to conceptual design that enables teams of designers to embedded semantic (S-B-F) information in their models. We believe that our approach offers several unique benefits: First, it is the Structure-Behavior-Function knowledge, more so than the geometry and topology, that encodes the designers' intent. By capturing this intent, we can search design knowledge-bases for related information and create pro-active design tools that can guide the search of the design space. Second, this approach liberates the designer for the usual restrictions of exact measurements, or precise positioning and orientation. CUP will aglow the designers' to create a 3D "freehand" sketch and the general structure of the artifact without detailed CAD.

- **Evaluation:** As a complement to our approach to conceptual design, we plan to undertake studies to rate efficiency of this design tool. How much is gained by having multiple users? What is the correlation between time efficiency and design accuracy in such environment? How do multimedia communication tools interact with this interface? What happens without them?

- **Internet Flux:** One technical challenge in this project was posed by the constant change in Internet, Web, and VRML software and technologies. Changes to development tools, Java, and browsers (Internet Explorer and Netscape) necessitated monthly updates to our development suite. Often the new tools were in conflict with older APIs and functionality.

  Our future plans are to port the current CUP to Java3D, a 3D programming environment for Java that supports VRML. We expect this to significantly reduce the integration problems among the tools, software, and browsers—and we expect to have fully completed the update by the time of this conference.

The immediate use for CUP is as a query interface to the National Design Repository. The structures created by CUP will be used to generate a query Repository and extract similar assembly designs. In It is our hope that this research expands the understanding of software tools can be developed to support conceptual design and lays the foundation for exploring new techniques to enhance our ability to search and retrieve 3D solid model data. Further, we believe that existing approaches to multimedia libraries can be augmented with geometric reasoning techniques that are tightly coupled with engineering knowledge and solid models—such as those developed in the future as part of this research.

Acknowledgements. Thanks to GICL members Jon E. John, who has maintained and extended CUP, and Yuriy Shapirshteyn, the resident GICL Java-NT-Unix-Web expert, for their assistance on this project.

This work was supported in part by a National Science Foundation (NSF) CAREER Award CISE/IRIS-9733545 and Grant ENG/DMI-9713718; additional support was provided by the National Institute of Standards and Technology (NIST) under Grant 60NANB7D0092 and American Telephone and Telegraph (AT&T) Laboratories, Internet Platforms Division.

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] S. Bhatta and A. Goel. Discovery of physical principles from design experiences. *International Journal of Artificial Intelligence in Engineering Design and Manufacturing*, 8(2), Spring 1994. Special issue on Machine Learning in Design. `ftp://ftp.cc.gatech.edu/pub/ai/students/bhatta/dp-aiedam94.ps`.

[2] D. Bowman and L. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, Rhode Island, April 27-30 1997.

[3] Thomas A. DeFanti, Maxine D. Brown, and Rick Stevens. Virtual reality over high-speed networks. *IEEE Computer Graphics and Applications*, 16(4):42, July 1996.

[4] Jose Miguel Salles Dias, Ricardo Galli, Antonio Carlos Almeida, Carlos A.C. Belo, and Jose Manuel Rebordao. Mworld: A multiuser 3d virtual environment. *IEEE Computer Graphics and Applications*, 17(2):55–66, March-April 1996.

[5] 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.

[6] James E. Fowler. Variant design for mechanical artifacts: A state-of-the-art survey. *Engineering with Computers*, 12:1–15, 1996.

[7] Bernd Frohlich, Martin Fischer, Maneesh Agrawala, and Andrew Beers a Pat Hanrahan. Collaborative production modelling and planning. *IEEE Computer Graphics and Applications*, 17(4):13, July-August 1997.

[8] Michelle Gantt and Bonnie A. Nardi. Gardeners and gurus: Patterns of cooperation among cad users. In *Transactions of the ACM Conference on Human-Computer Interaction*, 1992.

[9] A. Goel, S. Bhatta, and E. Stroulia. Kritik: An early case-based design system. In Mary Lou Maher and Pearl Pu, editors, *Issues and Applications of Case-Based Reasoning to Design*. Lawrence Erlbaum Associates, 1996. ftp://ftp.cc.gatech.edu/pub/ai/goel/murdock/kritik.ps.

[10] A. Goel and E. Stroulia. Functional device models and model-based diagnosis in adaptive design. *International Journal of Artificial Intelligence in Engineering Design and Manufacturing*, 10, 1996.

[11] Ashok Goel. Design, analogy, and creativity. *IEEE Expert and Intelligent Systems*, 12(3):49–56, May-June 1997. Special issue on AI in Design.

[12] Jonathan Grudin and Steven E. Poltrock. Computer supported cooperative work and groupware. *Advances in Computers*, 45:269–320, 1997.

[13] 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.

[14] Chris Johnson. Time and the web. In *Transactions of the ACM Conference on Human-Computer Interaction*, 1994.

[15] Jihie Kim, S. Ringo Ling, and Peter Will. Ontology engineering for active catalog. Technical report, The University of Southern California, Information Sciences Institute, 1997.

[16] Wolfgang Kruger, Christian-A. Bohn, Bernd Frohlich, Heinrich Schuth, Wolfgang Strauss, and Gerold Wesche. The responsive workbench: A virtual work environment. *IEEE Computer*, 28(7):42–49, July 1995.

[17] Wim Lamotte, Eddy Flerackers, Frank Van Reeth, Rae Earnshaw, and Joao Mena De Matos. Visinet: collaborative 3d visualization and vr over atm networks. *IEEE Computer Graphics and Applications*, 17(2):66–76, March-April 1996.

[18] James A. Landay. *Interactive Sketching for the Early Stages of User Interface Design*. PhD thesis, Carnegie Mellon University, School of Computer Science, December 1996. #CMU-CS-96-201.

[19] Valerie D. Lehner and Thomas A. DeFanti. Distributed virtual reality: Supporting remote collaboration in vehicle design. *IEEE Computer Graphics and Applications*, 17(2):13–18, March-April 1996.

[20] M. L. Maher, M. B. Balachandran, and D. M. Zhang. *Case-Based Reasoning in Design*. Lawrence Erlbaum Associates, Mahwah, NJ, 1995.

[21] Deborah A. Mitta and Patricia L. Flores. User productivity as a function of autocad interface design. *Applied Ergonomics*, 26(6):387–396, December 1995.

[22] A. Pang and C. Wittenbrink. Collaborative 3d visualization with cspray. *IEEE Computer Graphics and Applications*, 17(2):32–41, 1997.

[23] William C. Regli. Network-enabled computer-aided design. *IEEE Internet Computing*, 1(1):39–51, January-Feburary 1997.

[24] William C. Regli and Daniel M. Gaines. An overview of the nist repository for design, process planning, and assembly. *International Journal of Computer Aided Design*, 29(12):895–905, December 1997.

[25] K. Sycara and D. Navinchandra. Retrieval strategies in a case-based design system. In C. Tong and D. Sriram, editors, *Artificial Intelligence in Engineering Design*, volume II. Academic Press, July 1992.

[26] K. Sycara, D. NavinChandra, R. Guttal, J. Koning, and S. Narasimhan. Cadet: A case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2):157–188, 1992.

[27] David G. Ullman. *The Mechanical Design Process*. McGraw-Hill, Inc., 1992.

[28] R.C. Zeleznik, A.S. Forsberg, and P.S. Strauss. Two pointer input for 3d interaction. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, Rhode Island, April 27-30 1997.