masters thesis

a graphic user interface for a conceptual & collaborative design environment

by santiago v lombeida

dr. william regli, thesis supervisor

drexel university
philadelphia, pennsylvania

winter/spring 1998
Both, Conceptual and Collaborative design, are two of the fields in computer aided modeling which are growing in importance and feasibility as computer systems become faster, stronger, and growingly interconnected. We hereby introduce a study of a Graphic User interface for such systems; as we address different problems that appear as part of Interface Engineering.

**outline**

abstract
introduction 1.0
goals 1.1
topics to be covered 1.2
graphic user interface issues 1.2.1
conceptual design bases 1.2.2
collaborative interaction issues 1.2.3
theoretical background and research 2.0
conceptual design research 2.1
collaborative design research 2.2
hci and gui relevant research and topics 2.3
graphic user interface engineering 3.0
introduction 3.1
stages in gui engineering 3.2
requirements specification 3.2.1
interface design analysis 3.2.2
implementation and documentation issues 3.2.3
testing issues 3.2.4
prototype implementation and issues 4.0
requirements 4.1
functional requirements 4.1.1
non-functional requirements 4.1.2
design 4.2
data flow 4.2.1
structure design 4.2.2
issues on implementation 4.3
vrml issues 4.3.1
java language and awt issues 4.3.2
other issues and highlights 4.3.3
issues, conclusions, and future work 5.0
issues 5.1
conclusions 5.2
future work 5.3
acknowledgements
bibliography
structural graphical depictions :appendix a
screen shots :appendix b
source code and docs :appendix c
abstract

As computers grow faster, more reliable, and vastly interconnected, modeling systems are increasingly growing in potential for use in areas before purely done through direct human interaction. We are to study two such relevant issues in the field of computer aided design and modeling.

It is now clearly possible to have a group of engineers, architects, or designers scattered across large spatial distances, working on the same design file concurrently. Furthermore, intricate and complex case base reasoning systems are staring to allow these subjects the ability to design at conceptual level and then have the computer systems fill in/suggest the specific details based on large repositories of previously archived designs.

This thesis studies the requirements and possible functionality of a graphic user interface to be used for a computer aided design system supporting conceptual development in a collaborative environment. It opens the recent and seldom studied field of conceptual design to be accessible to end users without cutting edge computing resources, while interlacing it with ideas of a collaborative virtual design space, which itself has mostly been used along high capacity/bandwidth networks.
introduction

The engineering process of an effective Graphic User Interface is as much an art as it has always been a science. It is often the case that successful graphic user interfaces accurately survey and resolve the endeavor at hand, while encompassing easiness of use, simplicity, efficient use of the medium within which it is inscribed, as well as a basic aesthetic value.

The approach to creating such interface resembles that of any software engineering project. However, it is necessary to take in consideration special needs, such as intrinsic characteristic of user interfaces. Such needs will include the specific nuances and interaction psychology of the user and the interface inscribed within a particular environment.

Throughout this thesis we attempt to find an interface which fulfills two different needs in the computer aided design field: conceptual development and collaborative work. In order to create an interface with which allows for conceptual design one must first understand what is the process done for conceptual design outside the computer field, Furthermore, the need for the interface to work in collaborative systems add to the requirements the basic task of accurately depicting a collaborative space in which all participants may contribute in an ordered, precise, intuitive, and unambiguous manner.

This is the study of such creation process and methodology by proposing an answer to a Graphic User Interface for the use in 3D Modeling in its conceptual stage, in a collaborative matter.

goals 1.1

The objective of this thesis is to study and propose a model for a graphic user interface for conceptual and collaborative design. This study will take in consideration human-computer interaction issues in application to the relatively recent topics of conceptual modeling and collaborative work.

By the end of this thesis, we will be able to define what are minimum requirements needed for an interface with the given requirements, as well as be able to suggest different ways to approach them. It will also detail a description of one such interface designed for this sole purpose as a practical approach to the subject.

It is important to highlight that much of the functionality behind the interface is either currently available in some fashion (CSCW research,) or is currently under study and development (CBR in Design and Modeling.) Thus this study clearly focuses on the study and issues surrounding the graphic user interface for such system. However, as a final remark, since an interface is a window to the utility that lays behind, much of the study relies on the functionality and purpose of the systems themselves.
topics to be covered 1.2

Given the goals of this thesis, three main topics are directly relevant to this study. This topics are: graphic user interface issues, bases for conceptual design, and collaborative design issues. The first topic, issues on graphic user interface development includes basic Human Computer Interaction (HCI) principals, as well as issues directly related to computer aided design. The second topic covers the principals behind conceptual design, as well as a basic description of the approach which will be studied within this body of work. Finally, the last topic covers some pertinent information on Computer Supported Collaborative Work (CSCW) as it applies to the study at hand. These are described following.

graphic user interface issues 1.2.1

When dealing with functional graphic user interfaces it is important to keep several human computer interaction issues in mind: functionality, usability, and efficiency.

A graphic user interface the particular realm of design has to be able to offer the full functionality required by an expert user. At the same time, it is important that all this functionality be accessible to any user. Thus the basic principal of intuitiveness is basic to Human Computer Interaction (HCI).

Finally as users become adept to a system, interface interaction, the interface has to become as efficient as the user is, and as efficient as the user needs it to be. An interface used for design and modeling intents has to allow for user growth, and it itself be able to grow along with it.

As suggested by the classic papers “Post-WIMP User Interfaces” by vanDam [vanD] and “Non-Command User Interfaces” by Nielsen [Niel] new paradigms may be used beyond the traditional What-You-See-Is-What-You-Get (WYSIWYG) and Windows-Icon-Menu-Pointer bases systems to create user interfaces with greater potentials. Such new paradigms are introduced as part of Conceptual Design as object manipulation mechanisms, where the user may benefit from new interaction techniques.

It is assumed that the level the user dealt with, within the scope of this study, renages from what Ganti and Nardi ([Gant]) would refer as ‘gurus’ (field experts) to regular CAD users, which they also proved to have significant knowledge in the field beyond the average computer user. Furthermore, as Mantovani recognizes, it is necessary to establish a Social Context when dealing with HCI, specially when it is inscribed with collaborative work. Thus the setting encompassing this study is composed of american professionals with some degree of higher education, and knowledge in CAD modelling.

These variables will be taken in consideration and serve as basic assumptions (through functional and non-functional requirements) when it comes to designing a prototype.
conceptual design basis 1.2.2

It is a basic axiom of computer science that anything that can be done faster and more efficient by hand, has no need to have a counterpart in the computer world.

For instance, the basic idea of doodling or sketching something on a paper napkin serves only one purpose: the ability for the author/designer to informally materialize an abstract concept into paper. With such, she/he may be able to study it, stare at it if needed, analyze it, and quickly introduce new changes, or even discard the work without much being lost. The efficiency of a pen writing and rendering on paper without any delay or distractions is what makes this such an efficient process and thus a powerful tool in conceptual design.

Of course, if that was all there was to it, the conceptual design would be best left to napkins and the back of notebooks. However computers nowadays have the ability to interpret users sketches, match these elementary designs against previously created bodies of work, then suggest the author candidates as design alternatives, and even modify these candidates for the authors needs and intents. This is the concept behind case base reasoning systems, and thus the reason why would be worth it for an end user to “sketch and doodle” in front of a screen instead of on the before mentioned less formal mediums (as effective as they may be.)

It is important to highlight the close connection between the case base reasoning and the conceptual design tool. Furthermore, since a case base reasoning system may not only store the structural design, but functional and behavioral structure as well (know as SBF case memory structure,) then a conceptual design tool must have the ability to not only describe a design’s basic physical structure and hierarchy of components, but grasp further semantic information, in terms of the functionality or behavior of the design.

These represent some of the key issues in conceptual design that will be addressed by the proposed interface as described later in this study.

collaborative design issues 1.2.2

In the last three years much relevant work has been done in the area of collaborative design. Many of these latest efforts have been exhibited in Supercomputing Conferences supporting work across high speed/wide bandwidth networks virtual reality system. These studies represent a base reference upon which this thesis will base itself, upon in order to survey the current state of collaborative work in the design field. Some related topics which were addressed in these studies include different paradigms and protocols used to handle user interaction and collaboration in an shared work space.

However much of the understanding and knowledge we have on collaborative systems does not necessarily take place within the CAD field. Relevant work which will be taken
in consideration has roots in collaborative systems such as the one having been developed within Drexel University’s Problem Solving Environment (PSE) research teams. The name of this such project is TechTalk ([Laks]), a collaborative symbolic math program built to work on the internet, encompassing several Symbolic Math programs such as Maple and Singular. This project encompases issues in interaction between a user and the system, as well as studies in multi user interaction of people attempting to collaborate in scientific efforts across distances.

Other collaborative issues, such as basic representation of users within a virtual reality system, called ‘avatars’, are taken as field standards. Many studies of ‘multi user world’ representations and interaction have been realized, but are overlooked for the purpose of this thesis to focus on the design issues themselves.
theoretical background and research

All research conducted for purposes of this study centered around finding the current availability of conceptual design tools and technology, as well as researching on collaborative work used in the computer aided design work field.

conceptual design research 2.1

Conceptual design is the process by which a design is though of as schematized. This sketch of whatever is trying to be designed usually exaggerates those features which are most relevant to the design, as well as some few details which may make these design unique. For instance imagine a designer trying to deign a coffee mug. The designer may be sitting away from her workspace, doodling on the back of napkin. She would quickly render a quick semi spherical shape for the mugs body, as well as a curve on the side representing the handle. Similarly another designer allow drawing a coffee mug may decide to simply sketch a large cylinder representing the main body of the coffee mug, and then add a smaller cylinder hanging to the side representing the handle. Obviously in either case the designer has not paid attention to what the final designs measurements may be, what material, though it may be part of her/his internal brainstorming sequence. In both cases the user represented some quite different geometrical attributes (spherical vs. cylindrical) as well as a special feature (handle) which sets a coffee mug apart from a simple soda glass.

So the end user has spent a couple of minutes trying to illustrate some idea they had which has helped them clear in their mind what they will probably start drafting once they get back to their drafting board or computer screen.

In the last described example we have illustrated a case in which 2d sketching is used to conceptualize some design which is meant to be eventually constructed in three dimensions.

So is there a way we could do this 3d conceptual modeling directly? In fact there is, and they are utilized in some industries. Such include the use of clay models or paper and cardboard miniatures to represent something of volume.

The challenge in the computer science filed comes by trying to take advantage of this conceptual level of design to further develop some actual design based on its conceptual information

collaborative design research 2.2

Poltrock and Gruding [Polt] recognize that much has been done in the aspect of CSCW in the last five years. At the same time the recognize that because of its expanding nature and the fact that it has found its way into very diverse fields, research continues to grow, and some knowledge is just as relevant as its common ground with previous
work. This study will operate under this same understudying. Though much of work has been studied, the new applications taken in consideration opens a new undiscovered field to study and research.

Much of the current existing work being done on collaborative computer aided design revolves around immerse virtual reality systems. Work such as that being realized on “Collaborative Production Modeling and Planning” at Stanford [Froh] using the Responsive Workbench, or the “Collaboration in Vehicle Design” being studied at University of Illinois at Urbana Champaign [DeFa] using the Cave Automatic Virtual Environment (CAVE). The reason for this is because a environment where the user is able to dive into abets her/his ability to explore and maneuver around the objects in the environment, as well as to relate with other participants with a tactile virtual reality space. This however, when dealing with an immersive system requires high band-width networks as is studied in [Lamm] and [Sall].

Basic interaction among users has been well studied, and systems for collaboration are now available to build an environment on top of. Issues which come when dealing with collaborative design interaction are such as: object control, user visibility, environment control and view dispositions.

Issues in object control vary from the users ability to select an object, to a users ability to manipulate and modify an object.

Some of the research being conducted under object selection includes the ability of a user to select an object while in an immerse environment. In Bowman [Bowm] 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. In the first mentioned technique the user is able to “stretch” her/his arm to reach an object in space, while in the second she/he is able to cast a ray pointing to an object in order to select it. The first technique was tested to be harder on the user to actually get to an object, but once this was done the used could manipulate the object as if in her/his had. In the second case, casting a ray made it easier for the user to select an object, however because of the predetermined axis the degree’s of freedom to manipulate an object was cut back. In this study we will take advantage of this knowledge to take use of the given environment to create a manipulation paradigm equivalent to a hybrid answer as suggested by Bowman.

On the other hand research done on collaborative object manipulation was also studied by Pang and Wittenbrink in a system called Cspray [Pang]. 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 issues taken in consideration are such as “Two pointer Input for 3D Interaction” [Zele] in which it is suggested that two pointers may be used to manipulate objects in a more intuitive and efficient manner. Zelezniak and Fosberg, author of this paper, rely on the tests realized which 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.

These issues in collaborative design will be further involved as part of the actual interface modeling. Notice however that some common issues in collaborative workspaces (such as avatars) are taken for granted as part of the overall package.

**hci and gui relevant research and topics 2.3**

“One objective of computer-aided design (CAD) tools is to increase drawing speed and accuracy over manual drawing methods.” States Mitja and Flores [Mitt] on a appear on productivity in CAD systems. They explore how different interface designs and input methods have repercussions in CAD productivity. Their results in this study is not extremely conclusive towards one specific interface design. However strong points are made in favor in terms of more intuitive as well as customizable interfaces. They also purpose a method to study CAD productivity and efficiency in terms of test studies doing specific tasks. A similar point is also maintained in [Gant] in terms of CAD users being more productive when they have the ability to customize the interface for specific needs.

It is under this basis that this thesis takes to understand that the best way to create an interface to fulfill the prerequisites was to create a common base between CAD programs, customized for the specific purposes of conceptual and collaborative work. Furthermore, it is also understood to be that the final interface must be able to accommodate some kind of customization or ability by user to set the interface to suit her/his needs.

Other points which were taken into consideration and will be discussed in further detail are the different methods to display information (such as three dimensional data trees as in [Rohe]), handling information transmission delays through a network or web (as described in [John]),
graphic user interface engineering

When creating a graphic user interface the approach to be taken may be described in terms of a process which resembles that of a Software design, being one itself. This is the description of one such purposed method.

introduction 3.1

The modeling and implementation of a Graphic User Interface is as much a problem in software Engineering as it is in architecture and graphic design, and ultimately psychology (know as Human-Computer Interaction studies.) Thus a process needs be followed which keeps all issues pertinent, while serving is main goal as an interface (main functional requirement.) Thus this process must address the following issues:

◊ achieve its ultimate purpose of serving as an efficient window to the tool or data
◊ achieve such goal with an interface that is efficient and easy to use for the targeted user group
◊ maximize the utility and efficiency of the tool in the medium-environment which best fits its needs or the medium in which it is predetermined to be implemented in

With these points in mind, and based upon the Software Engineering process, the following scheme is suggested:

- Requirement Specifications
- Interface Design
  - Model Proposal
  - Model Testing
  - Model Refinement
- Interface Implementation
- Internal and External Documentation
- Testing

Notice that documentation must be implemented throughout the whole process, though it is depicted as one of the final steps.

This process is iterative as may be seen depicted in figure 1. An interface may suffer several cycles or revision
among requirement specifics to its model proposal and the actual implementation. Notice also that the interface design in itself is an iterative process of its own.

**stages in gui engineering 3.2**

Each of these steps enumerated have a counterpart within the software design world. However the considerations to be taken in place at each stage may differ, as well as may the different methodology to resolve each differ.

**requirements specifications 3.2.1**

The first stage of the Graphic User Interface life cycle entails a description of the interfaces specifications in terms of goals and requirements. Historically, requirement specifications in the software engineering world represents a description of the functionality required of the final product independent of design and implementation. This remains the same in the GUI world to a point. A Graphic User Interface in most cases represents a window to underlying tools and data. Thus it is intrinsically attached to this system/information base, and thus the implementation of a graphic interface may inherently attached to the system and its properties.

The requirements specifications for a tool as described so far 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 include exigencies not affecting the pure function of the object, but affecting other areas of the design (for example specific platforms under which the model is to run.)

In the later, non-functional requirements, specifics directly related to the functionality but delimiting actual implementation issues may be hidden so as to maintain the original software engineering ideals of software specifications.

**Interface design analysis 3.2.2**

This stage of the GUI engineering requires a very in-depth analysis of how to approach the different issues raised by the Requirement Specifications. The design itself of the interface is an internal iterative process. The first step is to propose a model. A model may be described in terms of input and output mechanisms, internal data structure and flow, as well as functionality and behavior.

After a model(s) is proposed it may undertake internal testing. This may be done independent of implementation through what is know as Wizard of Oz testing simulations. This means creating a mock interface in an environment such that it may be possible to study user interaction with the interface. For instance, in case of a virtual reality system with human avatars in it, a Wizard of Oz test may be done using human
actors interacting as the system would ([Niel]). In case of a WIMP interface this may be done with paper sketches, or even simple implementations using tools such as VisualBasic or even through the use of Java AWT.

This testing must have some kind of evaluation criteria with which the model may be judged and criticized. After such is done, the model must suffer some refinement. After this the model may be re-proposed and re-tested as many times as appraised needed.

**implementation and documentation issues 3.2.3**

After a model has been proposed it must be implemented within a selected (maybe required) environment. It is during implementation that some issues relating to the functionality may raise as unachievable. For instance, assume that a designed is trying to create a completely immersive system for study of shock-wave expansion. Such study would require that the system support fast rendering of complicated information representing the medium changes. Though it might be perfectly conceivable to the designer how something may be done, and even all tools that would make such environment optimal for shock-wave study; dynamic rendering may outside a systems hardware and software capabilities.

If such occurs, this different restrictions must be well document, as well as all efforts that go with trying to solve the problem at hand. If no solution may be reached within the current environment the designer(s) must re-evaluate the requirement specifics to either add the limitations to the non-functional requirements and thus accordingly change the interface proposed to offer new ways of reaching its goal, or change the requirements of the system to be able to handle the model as proposed (less likely to be acceptable.)

It is important to highlights to points obvious from figure 1. Implementation, based on a model proposal may in fact be an extension of the proposal itself. As models are being proposed there may be common ground which may be developed in order to test the models themselves. At the same time, depending of the environment in which a model was created in, some of what was purposed may be the basis or simply reused as part of the final interface. An example of the first is a simple event handling system which may be developed despite on how exactly the interface will do (given that it is a given that it is an event driven environment.) For the later, in case of WIMP systems, items like icons, menu design, and even buttons may have already been designed and maybe even implemented during proposal, and may be used as part of the final model. This also serves to highlight the importance of clean development and well document software engineering.

Second, internal as well as external documentation must be upkeep as the system is being developed. Internal documentation is to include all structure, data flow, input and output interaction (i.e. even handling and display interaction), not only on what is in the actual final version of the interface, but all venues studied and researched. This will enable future maintenance and improvements to know what has been done, why it was done the way it was, and what may done in the future.
testing issues 3.2.3

It is almost a given that when engineering software, implementation will take longer than what is scheduled. This is due to the fact it is practically impossible to account for issues only raised once the actual development takes place. That is why it is of most value to have created accurate model testing within the model design and analysis process.

Thus testing takes to faces. One is to test that the interface achieved its goals of functionality and usability, and the second is to verify that the product itself is consistent and works as expected.

Once the actual interface is in existence it must be tested for users ability to interact and understand. When doing this it is important to keep in mind the target group, and as outlined in [Gant] 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 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.

After a report is gathered where some measurement of acceptance may be calculated (though statistics on efficiency may be hard to gather in some specific fields [Gant]) the model may be revisited, and the interface design process may be traveled once more.
prototype implementation and issues

We hereby propose CUP (Conceptual Understanding and Prototyping) as an interface/CAD which allows the user to create entities with structural, functional, and behavioral information within a collaborative environment. The requirement specification for CUP are described as follows.

requirement specification 4.1

Conceptual design means the ability to model something without the need to define every detail. It means the ability to design at a conceptual, almost abstract, level. For example, if a designer wanted to refer to a coffee mug, it would be quite consuming to draw a hollow cylinder accompanied with a curved handle. It would be much more efficient if she was able to draw a simple cylinder with half a torus by its side; so that the computer system would be able to deduce her intentions, and suggest a couple of fitting designs. Now take this concept further, and imagine several people working across a network on the same design. That is what we mean by collaborative and conceptual design.

There are several research efforts which have created complex virtual reality collaborative systems across ATM and intranets ([DeFa][Sall][Pang]). Our focus is to bring these technologies to a medium widely available. Thus we have narrowed our focus to be platform independent and work on the internet (world wide web.)

Under this goals we propose an interface with the following characteristics:

- CAD based: the prototype to be created must have CAD capabilities for creation of designs/objects, with the added capability of embedding structural, behavioral, and functional characteristics

- VR environment: throughout the research conducted it became obvious the advantages of a Virtual Reality environment where the user has the capability of exploring a design as if working on a conceptual work bench [Rose]. Furthermore, the use of a virtual system adapts itself very well as a paradigm for collaborative interaction and work ([Lamm])

- communication: in order to create a multi-user environment, as well as be able to access a design repository, the interface created must have the ability to access some communication port

- portability: the interface to be designer must have the ability to work on any platform that has internet access and a browser (in the path taken, the browse must support VRML 2.0).

For our purposes and goals it is assumed that networking capabilities are available.
Furthermore we conceptualize 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 repository able to respond to queries based on physical (structure,) 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 (themselves irrelevant to this study.)

Specifics on this abilities as well as others are described following.

**functional requirements 4.1.1**

The tool to be designed must have the following characteristics:

- 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. This may be performed in a 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)

**non-functional requirements 4.1.2**

Non-functional requirements for this particular tool were dictated by the author to be:

- Requirements on the environment:
  - Should be platform independent
Must be developed for use in freely available software
Must be able to work across the internet

- Requirements of the interface:
  - 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 that may or 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 through the “eyes” of a different user

- Requirements on documentation:
  - There must exist extensive documentation on how to use the tool itself targeted for users of all ranges of expertise
  - Online help should be available, either as part of the workspace or as an accompanying internet site

- Other features that may be included:
  - Ability to create macros within the work space
  - Ability to set basic preferences on the working space

interface design 4.2

As a tool encompassing conceptual and collaborative work, this thesis proposes a virtual reality based system in which space is treated as a common. Furthermore, in order to keep in limits with the proposed user target group, and in contrast to the previous studies mentioned before, this tool is to work across the internet, using tools currently available freely. We will describe the model proposal, followed by detailed information on the input and output control interfaces, internal structure, and the functionality and behavior properties of the interface.

model proposal 4.2.1

Through the analysis of the functional and non functional requirements a basic idea of the functionality and environment in which the interface is to operate may be sketched out. The next task is to fill in the paradigms and concepts under which this interface is to operate.

The first challenge is how to create a conceptual design environment. In order to
answer this we must first research what is that we want to accomplish, as well as what is within our reach. From the previously discussed issues (see theoretical background and research) we conclude that what we want to create is a basic 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 objects, 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.

All objects will have a one to one relationship with a tag containing information on its structure, functionality and behavior. At the same time, two objects may be linked to each other (one or bi-directional) establishing structural, functional, or behavioral links between them. This also is done by a one to one relationship between a link and a link tag. Finally, a group of 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.

Thus, what this model proposes is an interaction between the three dimensional spatial word, governed by the objects, their geometrical characteristics, and their spatial relation to the rest of elements in the world; complimented by tree hierarchy/structure within the design which allows the ability of description of objects as part of a conglomerate, objects as independent entities, and objects as related one to another.

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 ([Pang]). 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 communications is being used (text chat, web phone, etc.), but at the same time users interact in the space by travelling the world, 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.

The private whiteboard or workbench allows the user to pursue design outside the shared workspace. This work may be later brought back to the shared workspace and contributed to the collaborative effort.

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.

Four different privilege levels of user may be established: administrators, privileged users, users, and guests. The difference between administrators and privileged users lie in the ability to deal with administrator level settings beyond the scope of this study.

At the same time we establish three different view modes which in term will set the ability of the user to access different views, tools, etc. These modes are: collaborative mode, search mode, and view only mode. The first enables users full use of the capabilities of the interface. Thus the required privilege is that of privileged user or above. The second mode, search mode, gives the user the ability to use a whiteboard and search the repository, but not to collaborate. The final mode is that of passive viewer, which only requires guest status, and that allows the user nothing but travelling the work space without any design privileges.

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.

interface mockups 4.2.1.1
Given these requirements and proposed solutions several mockups were created to graphically demonstrate what the interface may look like upon completion. Many details such as actual tools offered to the user will come to depend on the development environment used, and thus not demonstrated in these mockups.

We take the case of the design of a Military Urban Assault and Transportation vehicle. The fist mockup, figure 3, shows a user interacting with the environment though the Home view. Notice that other users may be seen as elements of this world.

The next mockup, figure 4, shows a free view of the environment without changing the specific location of the user itself. This also demonstrates the ability of the user to have selected a specific object (outside body) and give it a wireframe property, so that the internals may be seen.

The next two figures are based on the last one. Figure 5 shows the ability of the user to view information on a specific object on the design space. Figure 6 shows the designers ability to change the view yet again to see the object through a cutting plane (not available in first version of the product due to VRML capabilities restrictions.)
The next figure, figure 7, shows a top view of the world, still demonstrating all positions of the users. The last mockup, figure 8, shows again a top unrendered view of the world, but this time within a whiteboard space. This allows the user to create a platform (virtual floor) for the vehicle. Once the structure is created from basic design primitives offered from the top tool bar, the user may input the properties of the object, and if necessary perform a query on the shape.

Mockups were created using Adobe Illustrator, Adobe PhotoShop, and a trial version of 3D-Studio Max by Kinematics.

As these mockups were studied to actually detail an implementation plan several concerns were raised.

- These mockups display a level of design beyond what is required/expected of conceptual design. Though these mockups illustrate well points such as tagging, the reality is that the particular components expected of a conceptual design are not expected to be as geometrically sound as they were displayed.

- Links between components need be more obvious. When a relationship is established between two objects in the design, and thus data is embedded, this information needs be displayed in some for or other so as to convey the sense of component interaction.

**internal structure 4.2.1.2**

figure 7 – top view mockup

figure 8 – whiteboard mockup

figure 9 - view of layers in interface architecture
The internal structure of the interface may be devised as three layers, representing the communications layer, then internal data management and even handling layer, and the display layer. Information may flow across this layers back and forth from the user to the environment control management in the internal client itself and at the server.

The common thread that brings this layers together is a java applet (java application running within a www browser) which is responsible for creating sockets for communication with the server, up-keeping the representation of data, and communicating with the vrml browser.

Figure 10 - Event model
The communications layer is based on asynchronous communication received and sent to server, as well as to the internal representation layer and to the display layer.

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.

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.

**environment 4.2.1.3**

The interface is to be implement using the Virtual Reality Modeling Language (VRML) in congruence with Java and JavaScript programming languages as supported by VRML. Some standalone applications that support such include the free distribution package known as VRWave created in Sidney University, Australia. Also 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 and this be capable of supporting VRML. HTML will also be used for online documentation availability.

The environment used for development will include Pentium II based machines with Windows NT operating system as well as Sun Ultra machines with a Solaris operating system.

**Input and output interface 4.2.1.4**

The input and output capabilities of this interface is highly dependable on the environment in which is to be developed. The basic characteristics required are those as described in the requirement specifications.

As has been described so far, the interface will 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 will consist of the applet which will house all tools, and will be responsible for most of the interaction to and from the user. However, this apple will disguise itself as a simple collection of tools, but in fact will be the one responsible for handling all that exists in the VRML browsers as well as communication with the outside world.

The way this is accomplished is by invoking a set of classes from the applet, and then requesting for a handler for the browser from the internet browser. This interface between the VRML browser and a java applets is called the External Authoring Interface (EAI,) and should be supported by any VRML 2.0 supporting browser. The VRML EAI consists of a set of classes which are meant to establish a connection between VRML a browser’s and nodes’ (objects) events and the outside world. By
having access to a browser’s or nodes’ event, the EAI has the capability of modifying, adding, or deleting nodes, as well as handling changes or interaction among nodes within the VRML world. For further information into the EAI, it is described in the VRML Consortiums web page.

Now, because the EAI is the only window between the applet and VRML browser, the types of events that may be perceived from the VRML browser are limited, and revolve around objects/nodes themselves, contrary to the more complete event model available through Java. Such events include any changes in attributes (which may only be changed by calling the appropriate event) as well as events created by interaction with VRML sensors, such as proximity, planar, or spherical sensor. Through these sensors the applet will have the ability to tell when an object is being touched or dragged. Notice however that it must be the applet itself the one that actually moves the object by calling the correct events for an object, versus the object being dragged in the VRML world and then just reporting the action to the outside world.

With this understood, it is applets duty to offer the user the different capabilities as required. This capabilities include: creation, manipulation, destruction, tagging, linking, and grouping of objects. The VRML browser will be used as a display unit for the world (with all visual capabilities such as pan, fly, and zoom embedded in the browser) and to grab the selection events, as well as to handle (on its own).

**data flow 4.2.1.5**

Data flows across this interface as structures of objects and nodes. This information flows across the data structure according to actions take 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 denominated 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 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 intializer 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 a 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, this class is similar to that of a cone in that it creates a pointer between the objects link. A position method correctly places this cone (link), and is the method responsible for keeping the link in correct position if the from or to object linked are moved.

Finally the root and group nodes contain vector of object nodes. This hierarchy of classes is described in figure 11.

As objects (groups, links) get 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.

model testing and re-evaluation 4.2.2

As can easily be seen from the difference between the mockups and the final interface the interface went through processes of redesign and reevaluation. Most testing was done by the author, the supervisor, and a group closely related to the Design Repository and Geometrical and Intelligent Computing Laboratory (GICL) at Drexel University. Most testing was purely done on a basis of overview and feedback due to the fact that much of the actual work on the communications server for the repository and the case base reasoning system is currently in progress.

issues on implementation 4.3

Once the interface had reached a certain maturity level, which included a clear planning of data structure and flow, and the basic structure and input and output of the interface the implementation proceeded.

As the interface was implemented several of the issues tackled needed be reevaluated so as to be able to be constructed with the tools available. The actual implementation required in depth knowledge of VRML, VRML’s EAI, Java, and the Java AWT. Thus, we will focus on three different areas of expertise and work performed as part of the implementation: design tools, vrml issues, and java issues.

design tools and issues 4.3.1

The development and implementation of the design tools revolved around the capabilities available through the VRML Browser (Cosmo Player), and a Java Applet as a compliment. The VRML browser offers the user the ability to travel or examine the world. The Java Applet must therefore offer all other functionality. We divide this functionality into two categories building and manipulation tools.

Building tools are those whose function it is two create new objects, link them, group them, or delete them. Manipulation tools
are those which allow you to modify them.

Both these tool types are encompassed within the applet are, existing at the bottom part of the browser. The applet holds the tools together, though we offer the capability of detaching two of these group of tools. We discuss these as detachable tools.

**building tools  4.3.1.1**

The building tools are comprised of separate creator buttons for each of the different basic object shapes. This shapes are: boxes, spheres, cylinders, and cones. An object gets created by pressing on a button, which sends an event to applet. The applet then interprets this command, and sends a create new object to the VRML handler. Objects are added to root.

Once objects are created they may be selected, deselected using the pointer directly on VRML browser. Selected objects are highlighted in a yellow color. With an object highlighted the user may choose to delete it, which is accomplished by clicking on the delete button.

Users may also opt to link two objects together. This is done by selecting the link button, selecting the two desired objects, and then clicking on the link button again. Once the link is completed, a link tag will come up where the user may input all pertinent link information.

It is important to highlight, that in the linking process, no more than two objects may be selected at any time. If an object was selected prior to the link button, this object will be considered the source of the link. The last object selected is considered the destination. If all objects are de-selected, the linking gets canceled. In order for the user to know that the interface is in ‘linking’ mode, the information bar at the top of the applet is highlighted. In order to delete a link the user can simply select the link between the two objects, and delete it using the delete button (as used with objects). The link tag also offers a delete option.

Similarly, objects may be grouped by pressing on the group button and then selecting all object wishing to be grouped. Just as in linking, the info box will be highlighted for as long as the grouping process takes place. Such is shown in figure 13. Notice that for grouping purposes, more than two objects may be selected. Once all objects are selected the used must press the group button again, and the group get created. A group tag will come up requesting all pertinent group information.

In order to ungroup and object, simply select the user simply selects the group,
and then chooses the break bond button.

Through these set of tools, the user has the ability to create designs, and attach hierarchical and semantic information to the geometric structure as is displayed in the browser window.

**manipulation tools 4.3.1.1**

This second set of tools are those which allow the user to manipulate the objects themselves. The more straight forward tools in this field are the change color and tag tools. For the former one, once the user has an object selected, by clicking the color button, a color chooser screen will appear (if is already shown it will close) with the current objects color, and widgets facilitating color change. Meanwhile, the tag tool works similarly, displaying the selected object/link/group information tag.

Along with this we have introduced a more complex and dynamic paradigm for object manipulation in 3d space from a 2d interface. Much of the effort spent in this is due to environment user for development. Thus we must first understand the way objects/VRML nodes interact with the user through a browser.

As explained previously in this paper, objects within the world may be attached to sensors, which send event to the corresponding nodes, and thus may be captured by the browser. Such capturable events include the cursor moving over an object, the cursor dragging an object (the object does not get moved by VRML automatically,) and the cursor becoming active/inactive (mouse down/up.) The way out interface could take advantage of this is by allowing the user to select objects (objects get highlighted in a yellow color,) and then being able to drag them across the viewing plane.

Though this offers a level of functionality, it is far deficient as far as object manipulation from a CAD user is accustomed to. The CAD world resolves this usually in two ways. The first one is to allow the user to manually/numerically change the qualitative numeric vector (i.e. coordinate insert position) of a particular object. This approach is not intuitive but it is exact. Thus, when it comes to precise manipulating (i.e. positioning) is usually the most direct method.

The second way an object may be manipulated is through direct interaction with the pointing device. For instance in an immersive world, the user may literally grab the object and the twist it or move it to the desired position. In this environment much study has been done in different ways the user may select and manipulate the object to increase performance, relative intuition, and degrees of freedom. One such study is explained in [Bowm].

On the other hand, in a 2d display, the way most CAD systems approach this topic, is by offering the user multiple simultaneous view of the object (i.e. view ports/windows.) The user can then move a same object in different planes simply by manipulating in a different viewport. This is obviously far less intuitive than the immerse case, and it takes some time for the user to become familiar and thus effective in such environment.
In our study, we are using a 2d Virtual Reality browser. What this means is that we do not have the easiness to offer multiple view ports (though it may be done in the future.) However, we want to offer more than two degrees of freedom for the user to quickly manipulate an object. Thus we introduce the concept of a ‘move pad’.

The ‘move pad’ is based on the same paradigm used for travelling within a VRML browser, combined with the usability of a ‘touch pad’ (small rectangular touch pad used as a pointing device, specially common in modern laptops/notebooks.) A ‘move pad’ is a small viewport in the screen which represents a virtual coordinate system. Within the spaces described by this coordinate system the user is able to quickly relate to world systems, and from there manipulate an object.

Take the case of a user moving a window using a ‘touch pad’ the user is able to easily associate the movement in two dimensions of her/his finger on top of the touch pad and the movement of the window within the screen. The same way, a user may select an object and then use the ‘move pad’ to accordingly manipulate an object in any of the three base planes.

Figure 14 and 15 show two different versions of the movepad offered to the user. The first one shows three identical green rectangles where the movement/signaling may take place. The blue axis help the user discern which move rectangle is associated with which plane. The upper left rectangle represents the y-z plane, the upper right represent the x-y plane, and the lower right represents the x-z plane. The reason each of the ranges float off the axis, is to demonstrate to the user that the view system (move pad) is orthogonal, and parallel to the screen x-y axis. In other words, movement in the display screen in the x coordinate or the y coordinate (exclusive) will only map to movement in one dimension. Figure 15 displays the same concept, but maximizes the move areas, and labels each rectangle side appropriately.

The way manipulation actually takes place is by the user clicking down within a quadrant (plane), and then moving the pointer in the appropriate dimension. A line is drawn from the original down click point to the current movement point to facilitate visualization of the manipulation. The user may in fact achieve full degree of manipulation by simply choosing the manipulation mode (move, resize, or rotate) and then working within the move pad.

Further tests need to be done to prove the effectiveness of this paradigm, though initial tests have proven this tool to be straight forward, intuitive, and functional.

detachable tools  4.3.1.3
Both building and manipulating tools were deemed to be implemented as a detachable tool set. This was done by allowing the user to detach a set of tools with the click of a button. The tool sets that may be detached were two. The first, a group comprising of building and manipulation buttons, the second one the move pad.

Two buttons are offered to the user on right most part of the applet viewport. One for detaching/attaching the tool set (toolbar) and the second one to attach/detach the movepad. The image on the buttons change accordingly to represent the attach action (south pointing, red colored), or detach action (north pointing, green color). The tool set is symbolized by a pallet of shapes, while the movepad is represented with the basic axis and the corresponding quadrant areas.

The capability of detaching tool sets, allows the user to strategically place the tools on any location in her/his display, as well as resize this floating windows to offer larger views of the buttons or a larger view of the move pad. This not only may facilitate work and design, but it allows the user to control and adapt the interface to her/his best needs.

This is illustrated in figures 12 and 13, where the tools are attached in the former and detached in the later. Further studies need be realized to study the effectiveness to floating tools versus static tools. However, the benefit from resizing and dynamically repositioning a tool set was shown to be very promising during limited testing.

**vrml issues 4.3.2**

Virtual Reality Modeling Language has gained some level of acceptance in the last couple of years as a basic language for scene description. Since its original conception which included only scene description nodes, the creators realized that elements such as sensors could be used to embed information in objects, and facilitate their manipulation. But VRML is far from a powerful language that may be used for complicated modeling. It’s basic structure though very consistent in offer only exposed fields, events in or out, do not allow much in the way of scripting, or programming.

Currently there is a group of people that very strongly support the use of VRML as a base for software raging from CAD applications to games. And though the applications created in many instances are quite impressive, the future of VRML as an industry standard is nil. However VRML does hold a place, and a future, in the growing internet community. VRML is currently being used for creation of multi-user worlds that exist across the net, to 3d visualization of internet search results ([Rohe]).

Furthermore, in order to increase the potential of a VRML browser, an interface was created so that a VRML browser may be manipulated from an outside source, such as JavaScripts or a Java Applet. This interface is called the External Authoring Interface. However, due to its relative age, little documentation is found on the subject, outside the actual API, and small online tutorials which try to quickly grasp the larger concepts.

The way this EAI works within a Java Applet, is by the applet communicating with the
browser and the applet, and requesting a handle to the VRML browser. This is done with the `getBrowser()` method of the VRML Browser class. So what if such call fails? Well, that is the case some of the times, and there is not much that may be done. Why would such call fail? The reason a call like that may fail, is that the WWW Browser may have not fully recognized the VRML browser by the time the Java Applet makes the request. So how can this be solved? In the book by ???, it is suggested to incorporate code that talks directly with browser through JavaScript calls. A quite complicated solution which is not guaranteed to work.

It is a fact the software cycle followed by most companies rarely follow a strict Software Engineering standard, and most of the times is concerned with putting out the product as soon as possible. It is because of this that often it is hard to predict consistent reliability of a product. This is specially true in software related with the Internet, where the competition is fierce, and everyone is trying to be the first one to offer the newest release incorporating the latest technology.

Learning and dealing with the VRML EAI is quite a challenge. The following suggestions and issues where consistent throughout the development:

- The must be an underlying deep knowledge of the VRML structure, and the Java code developed must be done to mimic such behavior. In other words, if creating a class that will create a sphere, it is beneficial that it itself be a derived of a geometry class, which encompasses similar attributes (fields, event methods, etc) as the VRML nodes do themselves.

- The EAI description is the best reference available, and most of the times, the meaning of a method, or even the existence of a method may be derived from its analogous meaning in the VRML [language] itself.

- It is always good to try to foresee trouble happening with browser communication, but much effort must not be spent on this. Otherwise the answer may result in long complicated code, which may still result inept, and will most probably be bulky which in the internet is of great importance.

**java language and awt issues 4.3.3**

The power of Java AWT lies in its assertion that it will work in any platform that supports Java. Thus, many of the available components, are limited in their scope and actions. However components may me combined and modified to increase functionality, and to offer different components. However, due to the fact that this will be a composition of basic elements, and not a basic element itself, the efficiency may not be as expected, and behavior may vary from what is expected. This is the description of such proponents as developed to be part of the CUP interface.

There were five major components developed for use in the cup interface: picture buttons, widgets, the move pad, floating tool bars, and collapsible components.
picture buttons 4.3.3.1

Picture buttons represent a button which has the capability of having a picture instead of a label. Notice that as mentioned, it has the capability of holding a picture, though it is not necessary.

A picture button is first of all a panel. As such it may hold any component. In this case it holds just one component. This component may be an AWT button, or a picture frame. This is decided upon construction, and may not be changed. As a button, it propagates events such as mouse actions. This is easily achieved when the picture button is nothing but a simple awt button. In the second case, when the picture button holds a picture frame, we must first define what a picture frame is, and how it works.

A picture frame is a canvas which takes one or two images, and displays them at the appropriate time. This is decided by the user clicking on the canvas, which flips between the two images. When the user exits, or releases the button, the images are flipped again. The picture frame itself is not responsible for loading the images. This in fact, in our case, is taken care by the applets image manager (as defined by this author.)

Some of the challenges faced where to correctly display the appropriate image according to the size of the canvas. This is in fact determined by the layout component of the panel holding it. Thus when the paint command is called, the size must be recalculate, and the image displayed accordingly. Furthermore, when an event is taking place, such as a mouse down or drag, a repaint may be put on wait until events are finished processing. In our case, because we want the image to repaint once actions take place (since the image might have changed) the repaint must be forced. This is accomplished by directly grabbing (requesting) the graphics manager of the component, and then calling the paint command directly.

Another challenge is to correctly handle events. All circumstance must be taken in consideration. For instance a mouse down may happen without a mouse up due to a mouse exit. Similarly the mouse may be down with out a mouse down taken place due to a pressed pointer entering the canvas. All this must be considered, and propagated accordingly so as to portray button functionality.

widgets 4.3.3.2

In the case of the color chooser, it was important to allow the user to easily change a color, and be able to quantify the exact composition of it. That is easily achievable by offering to the user a slider/widget with which the user may handle the color compositions. This may be done with the use of scrollbars. However, scrollbars carry with them a connotation of moving a screen or a view port; and their size is unpredictable. This was solved by the introduction of a class called widgets which derives from a canvas.

A widget is a slider which may be set to track integers or floats, and range between a minimum and maximum value,
just like a scroll bar would. The difference is in the appearance, and the included rendering of the numbers representing the current value. Figure 16 shows the color chooser, which uses three widgets for the red, blue, and green numeric compositions ranging from 0 to 255.

The challenges presented in creating the widget are the event handling which must accommodate to grasp events which may mean changes in the widget (slider movement,) make those changes visible, and then propagate the events upwards. Just as in the picture button case, two major issues must be taken care of: direct re painting of canvas by requesting the graphics manager for the components, and correct rendering/resizing of the widget according to the canvas size.

move pad 4.3.3.3

The move pad is very similar in structure as the widgets previously discussed. In both its modes (as described in manipulation tools 4.3.1.1) the canvas which forms the pad must be take a point at which some action takes place, and make sense of it as a move in a certain quadrant, a mode change, a pad view change, or a null action.

Similarly, repaints must be done by direct calling, and in the case of action in of the quadrants, initial and past positions must be kept track of to correctly render the action line.

floating bars 4.3.3.4

The way floating bars were accomplished were to keep handles to the specific panels that hold each of the tool sets, and then keep handles of the tools sets themselves. While the applet may hold a tool set, such as the button bar (the set of all buttons as described previously) this bar/panel may also be held by a floating window, and thus detached from the applet viewport on the browser. This floating window is an instance of the class created named floating bar, which is derived from a AWT frame.

The main challenge encountered in this respect, was to correctly map events happening in a tool, whether it was attached or floating. This was done by mapping any event coming into the floating window, as if it was part of the main applet. This mean keeping a static copy (handle) of the main applet as a member of the floating bar. Once such was done, the applet would simply rationalize what kink of component was the event coming from (i.e. picture button) and thus take appropriate action.

collapsible components 4.3.3.4

When offering complex sets of information to a user, it is always best to find ways of grouping similar information together. Furthermore embedding such information such that the depth of the view of it is
controllable, increases the perception of the user towards the organization of this information, as well as saving space, and keeping it less crowded.

It is under this principals that it was deemed necessary that this interface include a way to embed components such that the components body may be hidden or collapsed to a single title to increase space use efficiency. Thus collapsible components were created.

The collapsible component is derived from a panel with a AWT Grid Bag layout, where the number of cells diverges between just one (in closed mode: the title), or two components (in open mode: title and component.) A thingy was created as a derivative from a canvas, which renders a small triangle indicating the components state. If the thingy is clicked then its appearance changes, and the event is passed to its parent which in turn changes its appearance/state as well.

Thus, the constructor for such a creature includes the component to be embedded, a title (optional), and a state (open/closed, optional). Furthermore these collapsible components are best laid out across grid bag layout panels of their own, so that as they change size, the panel holding them changes size.

**other java issues 4.3.3.5**

As the tools were implemented with the use of Java and Java AWT, several contingencies were raised. These included buttons not repainting when desired and button push not propagated as pointer is moved. The first problem was solved by adding validate, show, and paint commands to force an image to be shown accordingly. The second problem similarly, required a dissection of events relaying to a level up from the picture buttons themselves to the panel, and then relaying the event to the appropriate button.

In both cases the problem was completely unexpected. Furthermore, the problems may have been platform dependent, and in caused by the resource allocation mechanisms as implemented in the specific WWW browser used. We offer no further insight, and just highlight the large amount of work spent to solve such relatively ‘simple looking’ grievances.

**other issues and highlights 4.3.4**

The actual implementation of the interface was as much responsible for bringing forth new paradigms as was the modeling itself. Thus it is important to highlight the iterative cycle in which an interface is envisioned, planed, and engineered.
issues, conclusions, and future work

Throughout this study, we have been able to tackle one of the aspects in conceptual design, and at the same time introduce elements of multi-user collaboration, as well as a couple new paradigms for object manipulation and in interface design itself. We study the different issues arising from this, conclusions made, and the work that may follow from this study.

issues 5.1

Some of the issues raised as result of research, planning, and the actual development include:

- Studies such as that being realized between BMW and Dr. Peter Schroder from the California Institute of Technology approach conceptual design as a freehand sketching recognition problem. We have taken it to be 3d design with embedded semantic (SBF) information. Both approaches offer different building blocks in the aid of designers and modelers. The first makes design faster by recognizing what we had established to be more intuitive for a user as conceptual design, i.e. drawing on the back on an envelope. The second approach requires the user to in fact design using a computer without the usual restrictions of exact measurements, or precise positioning and orientation. The fact is that the recognition approach is one that extends beyond the field of CAD modeling, while the CUP approach focuses on working under the assumption of a case base reasoning supported Design Repository, and thus offers significant potential in use and efficiency. Is there some common ground? Are this two approaches complimentary, or just completely unrelated?

- From a more concrete point of view it seems that the state of software tools for the World Wide Web are growing in importance and power. However, at the same time, the ad hoc manner they are being produced takes away from their reliability. Netscape Browser, Sun JDK, and Cosmo Player despite that in the view of this author to be leaders in the industry all present a number bugs that would not be allowed in any other industry in the world. And in fact, as long as the industry keeps on going on exponential growth, many of these problems will not away. It is only our wish that as new products replace these old ones, less errors are introduce through more careful planning.

conclusions 5.2

The conclusions that may be made of this study represent a realization of the scope of this body of work.

- It is the author’s believe that CUP represents a valid approach to conceptual design.
• Furthermore, it brings multi-user collaboration in a Virtual Reality environment to a level which is functional and realistic.

• Several of the paradigms introduced for object manipulation are perfectly valid and represent a large amount of the effort introduced into this project.

**future work 5.3**

This interface and study offers one approach to conceptual design, however further work may be done on the interface, on the system behind it, and beyond the scope of both.

• The move pad was suggested as a paradigm from 3d manipulation in a 2d interface. Studies must be conducted to measure the effectiveness of such tool versus multiple 2d ports. Is there any benefit in having one port? Is there any way to improve the move pad for further efficiency? How useful is it?

• As a compliment to the conceptual design aspects introduced, collaboration was introduced. It would be beneficial to this and other studies to rate efficiency of design in a collaborative matter, versus individual work. 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?

• CUP is purely dependent on the assumption that a geometric design repository search and retrieval tool exists. In fact, this interface was created as a step-stone in this such project. As the project grows and the actual repository query system or CBR system is built, much study will be done in the most effective ways to catalog these design files, and thus the best ways to retrieve them. This means that work will then be needed to adapt this interface to respond to the needs of the repository. It is our hope that this interface serve that purpose well, and that work is continued to improve it.
Much gratitude is in order for all those which collaborated and supported this small piece of work. This includes thesis advisor, thesis editors, implementation collaborators, system administrators, and everyone that aided in one way or another.
structural graphical depictions : appendix a
screen shots :appendix b
source code and docs : appendix c