An Application Service Provider Infrastructure for Shared Workspaces in Internet-Based Collaborative Design

By

Jaime Solari

Submitted to the Department of Civil and Environmental Engineering and the Department of Architecture in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Civil and Environmental Engineering

and

Master of Science

at the
Massachusetts Institute of Technology
June 2000

Author _____
Department of Civil and Environmental Engineering/Department of Architecture
May 18, 2000

Certified by _____
William L. Porter
Professor of Architecture
Thesis Supervisor

Certified by _____
Feniosky Peña-Mora
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by _____
Roy Strickland
Chairman, Department Committee on Graduate Students
Department of Architecture

Accepted by _____
Daniele Veneziano
Chairman, Department Committee on Graduate Studies
Department of Civil and Environmental Engineering

Thesis Supervisor: Feniosky Peña-Mora
Title: Associate Professor of Civil and Environmental Engineering

Thesis Supervisor: William L. Porter
Title: Professor of Architecture

Thesis Reader: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

Thesis Reader Daniel J. Greenwood
Title: Lecturer of Architecture

An Application Service Provider Infrastructure for Shared Workspaces in Internet-Based Collaborative Design

By

Jaime Solari

Submitted to the Department of Civil and Environmental Engineering
and the Department of Architecture

June 2000

In Partial Fulfillment of the Requirements for the Degrees of
Master of Science and Master of Science in Civil and Environmental Engineering

Abstract

For architectural, engineering and construction projects involving transient 'virtual organizations' composed of non-collocated team-members, the adoption of concurrent design principles is seen as vital. An important aspect of concurrent design is the need for an effective communications infrastructure between team members. Traditionally, such communication has been handled through person-to-person meetings, however the complexity of modern projects has grown and as a result, reliance on new information and communications technologies is becoming increasingly necessary. Hence, within a concurrent design setting, there is the need for an integrated information and collaboration environment that will create a persistent shared workspace to support interaction between project personnel throughout all phases of the project. This research explores computer-supported mechanisms for enhancing distributed design collaboration. The goal of this thesis is to develop a set of requirements, system architecture and an early system prototype to facilitate computer-supported collaboration among distributed teams. The prototype will consist of a persistent shared workspace system built from the integration of complementary collaborative applications. These applications are the CAIRO system, developed at the Massachusetts Institute of Technology and the VNC system developed at the Olivetti Research Laboratory.

Thesis Supervisor: Feniosky Peña-Mora
Title: Associate Professor of Civil and Environmental Engineering

Thesis Supervisor: William L. Porter
Title: Professor of Architecture

ACKNOWLEDGMENT
May 24, 2000

I would like to thank Nancy Jones, who has made it possible for me to be here and to Feniosky Peña-Mora, who has made it possible for me to stay. Thanks to the faculty members involved in this thesis: William Porter, Jerome Connor and Daniel Greenwood. Special thanks to all the members of the team in its various incarnations from DISEL to 888 and beyond: Kiran Choudary, Gyanesh Dwivedi, Chang Kuang, Justin Mills, Sen Sugata, Sanjeev Vadhavkar and Padmanabha Vedam without whom this project would not have been half as much fun. Acknowledgements for the people that have provided open source code and helped avoid 're-inventing the wheel'; John Wilson and the VNC project deserve special mention. Finally, I would like to thank my friends and family for their non-collocated support over the past years.

Para Lela y Darío

Contents

List of Figures

All illustrations by the author except those noted otherwise.

Chapter 1

1 Introduction

Construction projects usually involve transient 'virtual organizations' made up of members of a project team working together on the design and construction of a facility. Team members are often non-co-located, particularly at the early stages of the design process, and tend to work independently while making decisions that affect others. For these cases, the adoption of concurrent engineering principles by the construction industry is increasingly being seen as vital for reducing the problems posed by the industry's fragmentation, and enhancing its competitiveness. An important aspect of concurrent engineering in construction is the need for an effective communications infrastructure able to transmit project information between members of the project team and across all stages in the constructed facility's lifecycle [Anumba and Duke, 1999]. Traditionally, such communication has been handled through person-to-person meetings, however an increase on the reliance upon information and communications technologies has shifted this project information online. Hence, within a concurrent engineering

setting, there is the need for an integrated information and collaboration environment that will create a persistent workspace to support interaction between project personnel throughout all phases of construction projects.

As these information and communication technologies move rapidly into the world of networks and distributed environments, architecture engineering and construction companies are looking forward to the wealth of opportunities that are being uncovered with this new form of collaboration. By combining the computing power of today's processors with real-time data transmission, virtual collaboration forums over distributed networks will become a powerful tool in tomorrow's workplace.

This research focuses on creating those persistent workspaces using new models of computing delivery in conjunction with an understanding of group meetings and collaboration processes to facilitate computer-supported collaboration among distributed team members. This initiative explores computer-supported mechanisms for enhancing distributed engineering collaboration by using the Application Service Provider (ASP) model to create a shared workspace for computer supported collaborative work.

This thesis starts by outlining in Chapter 2 the most important features of the ASP model. In Chapter 3 a requirements analysis for a collaborative system is provided. Following the requirements, Chapter 4 discusses the concepts behind shared workspaces for CSCW. Chapter 5 explains the methodology behind this research and the two strategies pursued. Chapter 6 outlines the most important features of the CAIRO system developed at MIT and the Virtual Network Computing applications developed by the Olivetti and Oracle Research Laboratory. Chapter 7 describes the design considerations for a Collaborative ASP. Chapter 8 provides an overview of the benefits of an integrated collaboration and shared workspace environment. Finally, Chapter 9 is the summary and conclusions for this project.

1.1 Motivation

Significant research efforts have been devoted to the area of sharing information through computers and the Internet. However, limited attention has been devoted to the basic communication mechanisms and the encoding of these mechanisms [Peña-Mora et al., 1997].

Therefore the objectives for the integrated system described in this thesis are three fold. First, it aims to remove the same-place constraints that are characteristic of face-to-face meetings. Elimination of same-place requirements allows for true global collaboration without collocation, minimizing the overhead associated with collaboration as well as reducing project expenses.

Second, in addition to the removal of the physical limitations of in-person meetings, the integrated system also seeks to remove the temporal or same-time constraints. Without same-time constraints, participants could contribute to the collaboration process asynchronously, adding convenience to the entire collaboration process.

Finally, the system seeks to model meeting control structures. Since limited attention has been devoted to encoding these basic communication mechanisms, the system described below implements many forms of meeting control structures, in order to facilitate the flow of information.

Keeping in mind the motivations outlined above, the goal of this research is to develop a set of requirements, a system design or architecture and an early system prototype that will meet the requirements. The prototype will consist of a comprehensive working collaborative system built from the integration of collaborative applications. These applications are the CAIRO system, developed at the Massachusetts Institute of

Technology and the VNC application developed at the Olivetti and Oracle Research Laboratory.

1.2 Objectives

A working prototype is intended to demonstrate the collaboration enabling capabilities of a meeting environment in combination with a persistent shared workspace within the context of an architecture, engineering and construction (AEC) project. This prototype will employ the meeting control structures, which is central to the original CAIRO system in combination with a new model of application sharing deployment. A sample scenario will be created to simulate the types of situations that may occur during a design meeting. These sample scenarios are intended to provide insight to the various kinds of interaction that need to be managed during collaborations. From this example, an architecture that incorporates the new application sharing model and the underlying database design is developed.

From the work described above, a robust collaboration system is shaped, allowing for more effective shared workspaces to occur, both among distributed team members and within independent isolated members. In other words, participants may choose to work as part of a collaborative group or alone in their own private workspace. The collaborative ASP model incorporates additional CAIRO features such as full hierarchical group structuring, agenda building, and a detailed information policy to facilitate this interaction process.

1.3 Hypothesis

The architecture for the new application collaboration system will incorporate important concepts of the current CAIRO system. In particular, CAIRO possesses concepts that relate interaction norms in a meeting to a virtual environment, such as the

control structures. However, the ability to share information beyond a whiteboard needs to be incorporated to allow for comprehensive application and data collaboration.

By developing and using a application sharing prototype designed and deployed with the ASP delivery model, experience can be obtained for generating a roadmap for next generation systems. Based on the advantages and disadvantages of the prototype, a new set of requirements can be developed that define needed changes.

1.4 Benefits of This Research

The successful development of this collaboration system will greatly enhance the effectiveness of concurrent engineering teams where members of the design team, suppliers and consultants cannot all physically meet at the same location. First of all, this new system will enable team members to collaborate through a shared workspace from multiple platforms and operating systems, which is currently not possible with most collaborative tools. Second, through its thin-client architecture, the system will enable designers to work on applications even when the application is not installed on the local computer or belongs to another platform. Third, through its database connection, the system will enable to designers to query the sessions for design process information or conflict resolution. Fourth, by taking advantage of current developments in mobile computing, the system enables team members to collaborate without the need of desktop computers. Designers and other team members can work on common documents through Palm Pilot and Windows CE devices. Team members will be able to collaborate with each other from locations wherever their physical presence is most valuable (i.e. on-site or at the design offices). Thus, dealing with unexpected issues from the construction site or the design office will become easier. In addition, it will be feasible for associates to participate in multiple meeting sessions simultaneously. This way, consultants and experts can contribute their value to more organizations than they have in the past. Time, monetary resources, and lack of involvement do not have to be compromised.

Chapter 2

2 Application Service Provider

Multi disciplinary teams working in a distributed environment are very often working in a heterogeneous network, with different computing platforms, namely Unix and Windows. Also, as wireless technology comes of age and mobile devices become commonplace, the platform compatibility issue will be combined with the hardware/interface compatibility issue. For example, as well as having a network with Unix and Intel platforms, there will be handheld devices coexisting with mobile phones and the software will need to take this into account. This compatibility problem is inherent to traditional client/server architecture, which emphasizes client-side computational power [Citrix, 1999]. One solution to this problem is server-side computing which is used by the Application Service Provider model.

2.1 The ASP Model: A Brief History of Outsourcing

In the early years of business computing, outsourcing was known as "time-sharing." Expensive and complex mainframe computers were beyond the reach of all but the largest corporations. Consequently, many businesses shared the processor time of an off-site mainframe that was managed by a third party. This time-sharing, or "bureau," service typically rented the mainframe for relatively easy-to-manage tasks, such as payroll processing or receivables billing. Mainframes were used mostly for number intensive data processing rather than managing complete applications. When affordable minicomputers and personal computers arrived on the scene, demand for time-sharing dropped, and many processing tasks were brought in-house.

Time-sharing was not scaled back because it was a bad idea, but because PC technology reduced the costs of processing tasks internally. The practice of outsourcing tasks that are not part of an organization's core competency remains a sound one. However, today's business environment calls for timely, interactive access to applications and management information, something time-sharing lacked. Until the Internet came along, there was no cheap, nonproprietary means for a desktop computer to communicate with an off-site application host system and no standard client environment that could interact with the remote application.

The Internet provides a publicly accessible infrastructure that connects users to off-site application servers. Desktop Web-browser software provides a standard way to interact with an application hosted on a Web server. And neither the Internet nor browser software adds much cost to the outsourcing equation. Using this technology, software vendors and Internet Service Providers (ISPs) are reviving outsourcing for information management. Software vendors are providing "rent-an-application" services, and ISPs are expanding their business models to differentiate themselves in what has become a highly competitive environment.

2.2 What Is an ASP?

Conventional ISPs manage Web servers and e-mail servers that are connected to the Internet. These servers host Web pages for businesses and individuals and route e-mail messages. An ASP simply extends this model to include software programs, from a payroll or human resources module up to a full enterprise resource planning (ERP) suite. In addition to hosting Web pages and e-mail, ASPs use servers connected to the Internet to host applications. ASP customers can interact with a remotely managed application module or suite of modules via Web-browser software on an anytime, anywhere basis.

Simply hosting the application software remotely is only part of the job of a full-service ASP. The ASP has to perform a role that combines the responsibilities of an ISP, a traditional outsource service provider and a value added reseller (VAR) from which you might have purchased a non-customized software application. In the near future, more ISPs will become ASPs; ISPs will partner with software vendors and VARs to offer ASP-like services; and vendors and VARs will simply become ASPs. Buying prepackaged applications for in-house use, rather than renting them over the Internet, may become a thing of the past [McKie, 1999].

2.3 ASP Outsourcing Benefits and Drawbacks

Outsourcing something like an ERP application to an ASP has many benefits that apply to any type and size of business. The ASP maintains the hardware server "farms" (large facilities of servers) required to efficiently host complex applications and removes the need for companies to buy, maintain and upgrade in-house hardware. The ASP can make sure that the latest versions of applications are available to enterprise-wide users without the need for costly site-by-site in-house upgrades. Using an ASP-based ERP system also means that the only client software required on the user's desktop is a Web browser, which eliminates the need to manage client software on a desk-by-desk basis.

IS cost reductions are not the only reason why ASP outsourcing is attractive: ASP customers can sign up new users or workgroups for an application at almost a moment's notice without the need for complex infrastructure and implementation-resource planning. New users can access the application without expensive upgrades to the local technology environment. This means businesses can get new applications, such as sales force automation or customer relationship management software, up and running faster; can bring on more users or users from remote offices more quickly; and can adapt more easily to merger and acquisition activity. Small businesses and geographically dispersed multinationals alike can capitalize on the benefits of application service providers.

However, there are some clear drawbacks to the ASP outsourcing model. Switching from an internally managed and accessed local- or wide-area network (LAN or WAN) to the publicly managed and accessible Internet means that access to outsourced applications may be subject to influences beyond your control. For example, heavy Internet traffic may slow application response times, and malicious hackers could get hold of accounting, employee or customer data. Furthermore, not every application available today has a complete or thoroughly field-tested Web interface, something that could restrict its availability to users. Also, a complex system such as an ERP suite requires considerable time to configure to a company's specific business needs, and integrating an ASP-managed ERP system with complementary in-house systems, such as a customer relationship management system, could prove challenging.

2.4 ASP Technology Requirements

While the technology requirements for running an outsourced application over the Internet are daunting for the ASP, they are relatively straightforward for the application user. Depending on the design of the application being outsourced, your company should need no in-house application servers or database servers to support the application. Similarly, if the ASP is hosting a properly designed browser/server application, it should demand nothing more than a Web browser on each device (PC, laptop, handheld, mobile phone, etc.) that needs access to the application. Clearly, every user of the outsourced

application requires secure access (via a firewall) to the Internet and, ideally, should have a full-time, high-speed connection using a virtual private network (VPN) managed by the ASP. Users also must have e-mail since this is how the outsourced application delivers reports, documents and business alerts, facilitates workflow participation and maintains a support dialogue with individual users. Given the fact that most businesses already have access to the Internet and e-mail, they won't require much new or even upgraded technology to take advantage of an outsourced application [McKie, 1999].

2.5 ASP Applications

A recent survey of the ASP industry showed that communication and collaboration applications would be most frequently deployed using this model:

Question. In the year 2000, which class of applications an ASP will most frequently deliver?

33% - E-commerce

25% - Communications/Collaboration

15% - Customer Relationship Management

14% - Finance/Accounting

7% - Human Resources

6% - Education/Training

[Online survey, ASP Industry Consortium 1999]

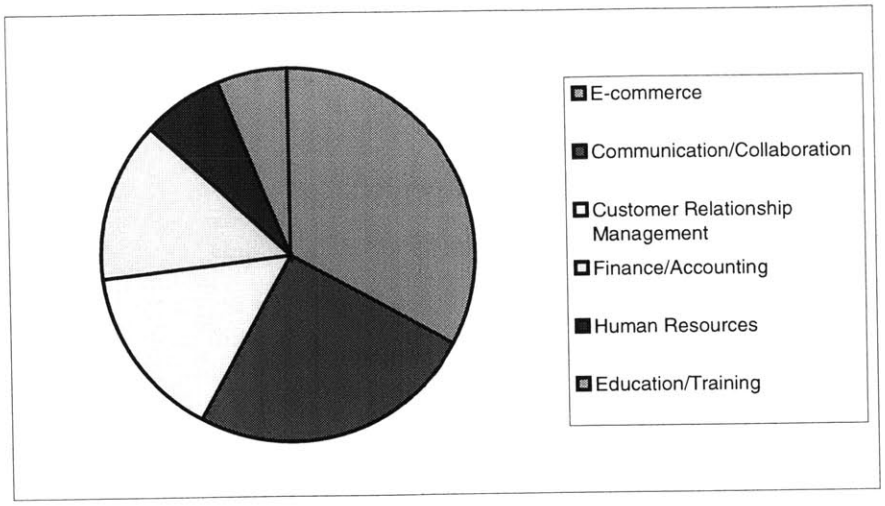

Figure 2-1. Which class of applications an ASP will most frequently deliver?

As it will be explained in the requirements, distributed design teams often use different software components on heterogeneous platforms and networks [Figure 2-1], therefore it is important that they are interoperable to allow collaboration and data sharing. With server-based computing, applications are deployed, managed, supported, and executed completely on the server.

Due to the inherent advantages of server-based computing and the distributed nature of the virtual design teams, the deployment through an Application Service Provider model would be of great benefit to enable collaborative applications. These benefits can be summarized in the following three critical components:

A. A multi-user operating system that allows multiple concurrent users to log on and run applications in joint as well as separate, protected sessions on a single server.

B. A remote presentation services architecture capable of separating the application's logic from its user interface, so that only keystrokes, mouse clicks, and screen updates travel the network.

C. Server-based computing does not require applications to be downloaded to client devices. As a result, application performance is neither bandwidth- nor device-dependent.

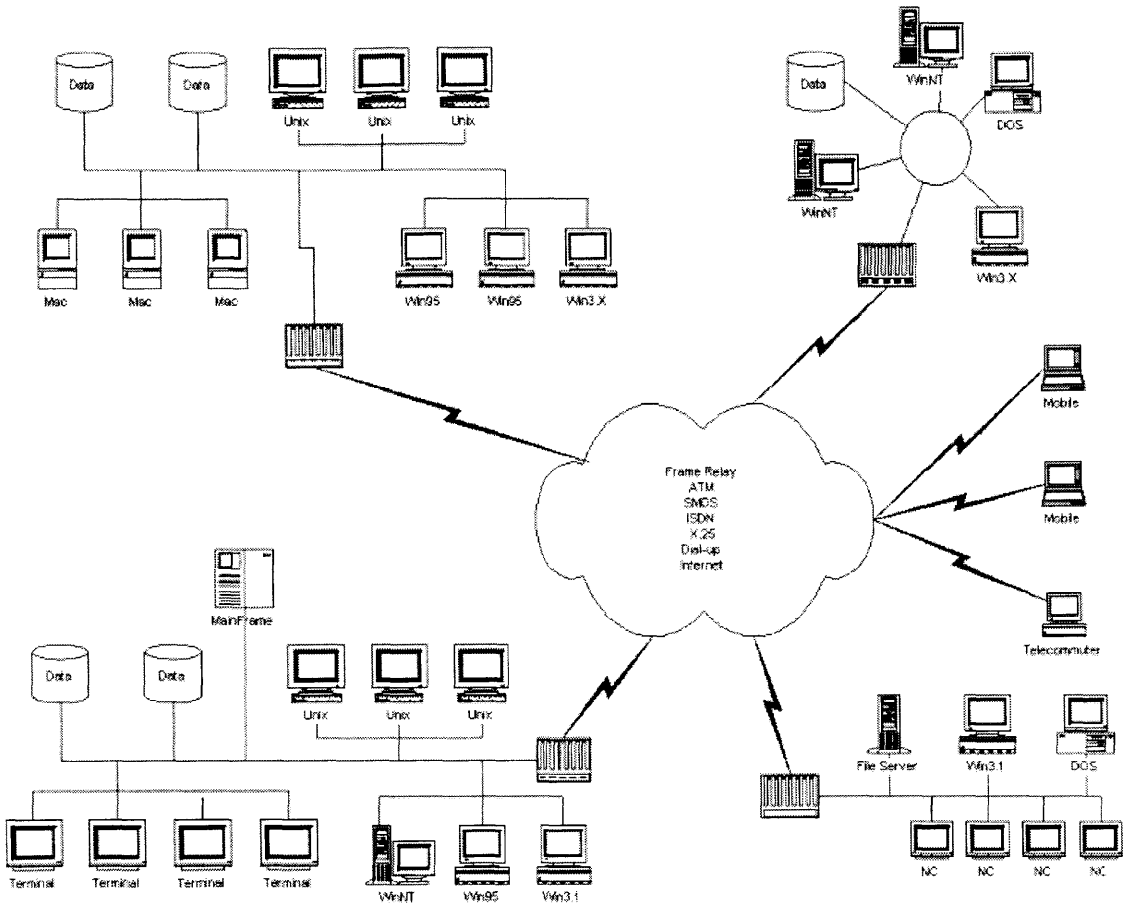

Figure 2-2. Heterogeneous Computing Environments [Citrix Systems Inc. 1999]

In summary this Application Service Provider model can be used for the application sharing implementation within a shared workspace. Shared applications can be easily deployed from a server instead of running on the local client machine and this way avoiding a number of problems. This new model of deployment solves a diverse set of challenges that will enable to fulfill the requirements explained in the following chapter:

A. LAN-Locked Applications. Applications based on two-tier client/server architectures are designed for the LAN and are not optimized to run over high-latency phone or WAN connections that run 100 to 1000 times slower than a local segment.

B. Heterogeneous Clients. Networks usually involve PCs as well as non-Windows systems such as Linux, OS/2, UNIX, or Macintosh. Other networks could include low-cost, fixed function devices, such as terminals or wireless devices such as wireless tablets and personal digital assistants (PDAs).

C. Management. Managing access (security), version control (maintenance), system configuration (moves, adds, deletes), and support (help desk) can be very costly particularly for distant users.

D. Heterogeneous Software Packages. Each subgroup within a virtual team typically uses a set of software packages that is specific to the group's discipline. Therefore compatibility becomes an issue for situations such as visualization.

2.6 Summary

The first part of this chapter described the typical components of the traditional ASP deployment model that is currently being used in industry. The second part of this section described the attributes of an ASP deployment that would be of benefit for a shared workspace. The reasons why a shared workspace would benefit from an ASP model are formalized in the following chapter under the requirements.

Chapter 3

3 Requirements

Before reviewing the requirements of the new shared workspace it is important to outline a scenario of how this shared workspace would be used. This scenario describes the behavior of a generic collaboration tool and within its features the shared workspace is highlighted.

3.1 Scenario

The following description is a user session of an integrated collaborative system that involves a building design problem [Peña-Mora, Anumba, Solari, Duke 2000]. The parties involved in the design are a project manager, an architect, a structural engineer, an environmental engineer, a contractor and a geotechnical engineer. A member of the design team wishes to alter the dimensions of a particular area of the building. They see

that this impacts upon the available corridor space and feel they should check that no safety regulations covering the size of corridors are contravened. They wish to contact an individual in the virtual organization who will be able to help them with the issue. Not knowing the name of such a person, they can look up in a project repository the person that would be responsible for those aspects of the project. The project repository contains all relevant information relating to the facility being developed including details of which disciplines are interested in specific aspects of the facility.

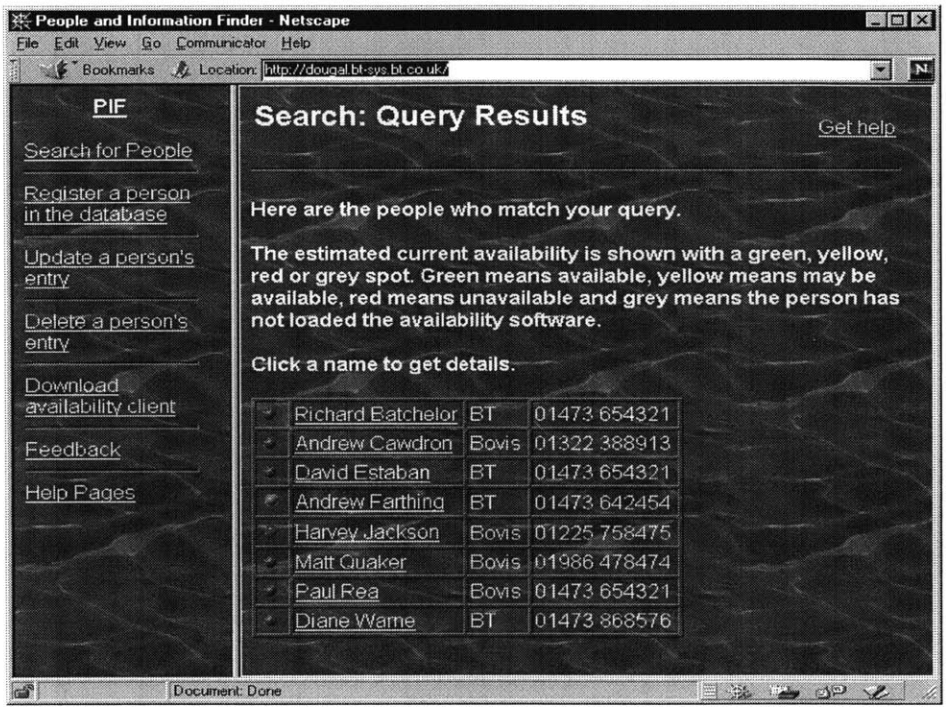

Figure 3-1. Query Results Page [Anumba and Duke 1999]

Once the people affected by the decision have been determined, the system will determine the current availability of each member involved. People in the result set might have a colored dot next to their name to indicate the result of the availability look-up (For example green means they are probably available, red means they are unavailable and gray means that the system has no record of that person). The user can then see, at a glance, which of the people are available and select them from the list. This selection returns more detailed information from the database, such as images from a 'Web cam'. The image from a camera pointing at the desk area of the individual is periodically

written to a file. This is then added to the web page. If the required person is available, communication can be launched automatically from within the system, with the project model providing the context for the design discussions.

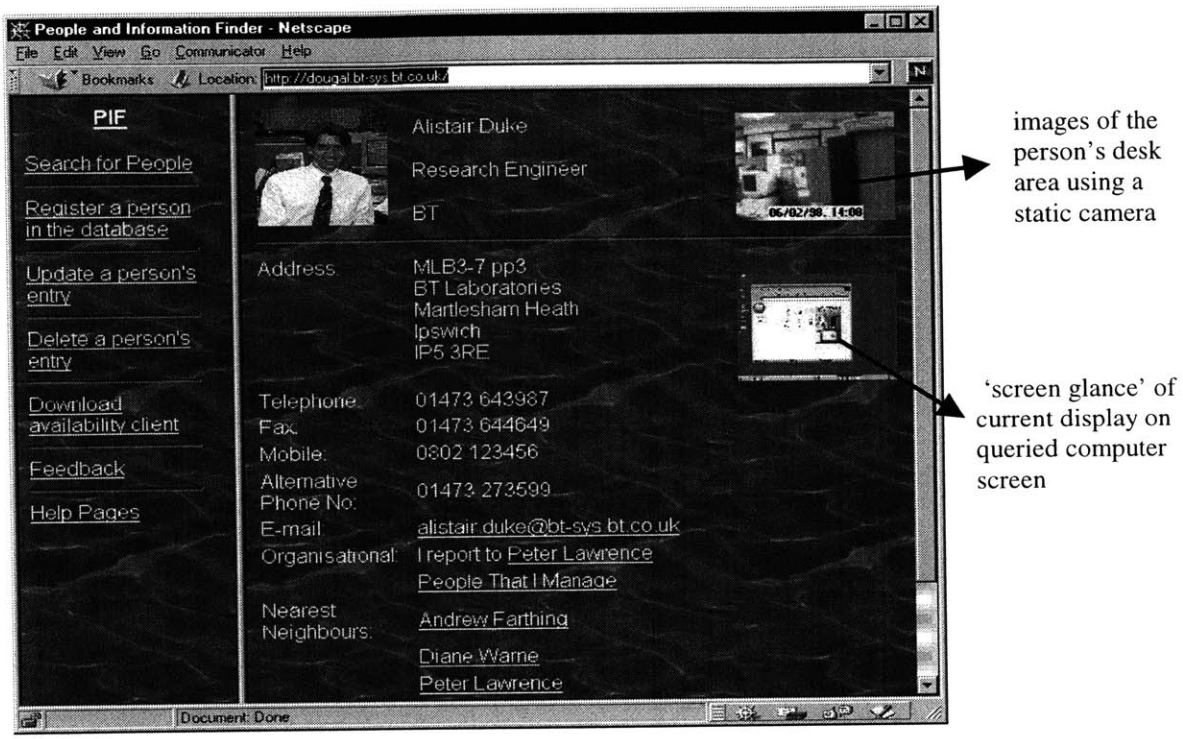

Figure 3-2. Personal Details and Availability Page [Anumba and Duke 1999]

At the same time that communication links have been established, a graphical representation of users clustered by disciplines, activities, interests and projects in a multi-layered environment can be provided. Close to the user's avatar are other members of the virtual team. However the user can navigate freely about the world, and communicate with other people if desired. The user starts up a text chat conversation with a member familiar with safety regulations but moves on to more important issues and decides to upgrade the conversation to a voice chat. The two avatars are selected and automatically entered into a phone call together. The other people in the space have a visual indication that you are in an audio-chat, and can overhear some of the conversation as it is 'streamed' into the world. Another colleague moves into the 'audio zone' and is automatically phoned-up to enter the conference. He has overheard talking about the

particular project in which he is involved and would like to discuss a certain aspect of it [Anumba and Duke 1999].

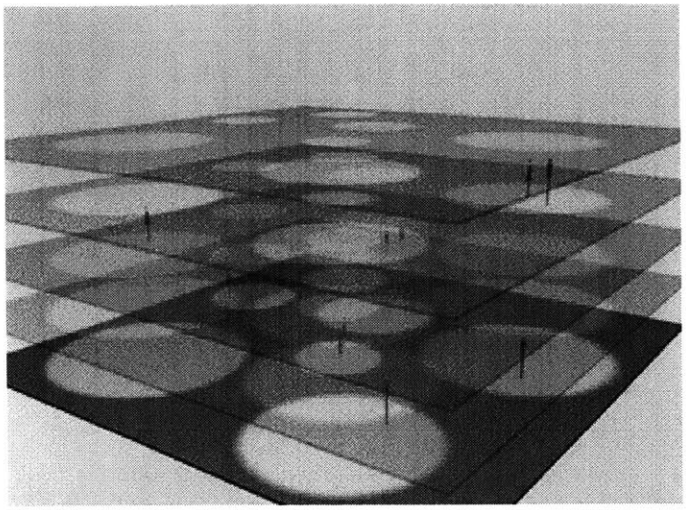

Figure 3-3. Graphical Representation of Users clustered by disciplines, activities and interests [Anumba and Duke 1999]

All three members decide to continue the discussion in private and perhaps take some notes, so they choose to enter a more formal meeting space. While they continue chatting, their avatars are moved up to the formal meeting plane and a meeting room appears around the avatars. Various conferencing tools appear such as a table and whiteboard and the avatars start to 'act out' the interactions between the team members. The project manager initially selects a 'freestyle' forum style from the available choices. The forum moderator then requests additional information regarding the meeting, these include: media drivers used (text, audio, video and whiteboard); maximum number of active speakers (2); and membership criterion (predetermined list). Additional invitations to join the meeting can be sent by the members currently attending the session. The members of the design meeting can setup a meeting agenda helped by a wizard and discuss the issues related to project as they each formulate their evaluation of the problem. Eventually, this meeting could become confusing as members continually interrupt each other's work. At this stage, the structural engineer may wish to revise the meeting moderation scheme to a 'chairman' scheme. The structural engineer can do so by entering the control parameters he/she wishes to enforce on the members and the meeting

would be reestablished with a chairman control scheme. The architect raises his/her hand to present a design proposal to the other members and receive a critique from each, so the architect is given the floor. Eventually through multiple cycles of discussion and explanation the design would be formulated with acceptance from all participants of the meeting. The negotiation would be facilitated by the social feedback provided by the awareness driver as well as the expressions tool.

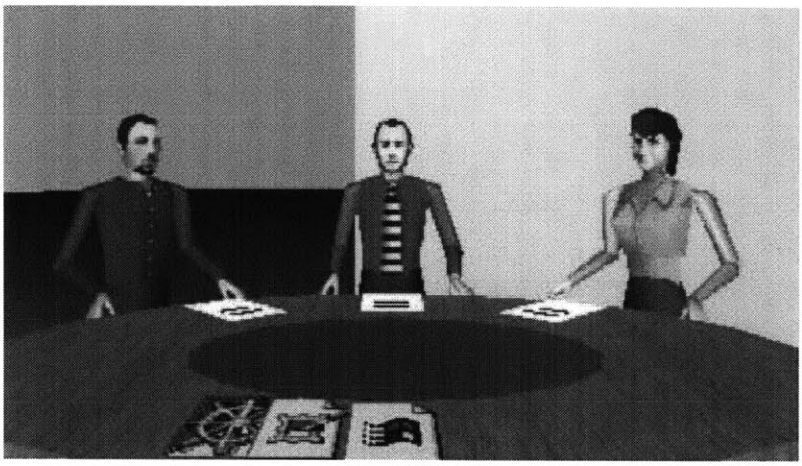

Figure 3-4. The Conferencing Room [Anumba and Duke 1999]

Next, the safety expert decides to present an overview documentation for the task he/she wants to discuss. The safety expert enters the details of the CAD document on the local desktop, and the document, which had been private until this point is shown as a paper on the table in front of the other avatars, to represent its public accessibility, with the CAD icon on top. Any meeting participant can now select the paper and the document appears in a 2D window on the local screen, even if they don't have the CAD package installed on their computer and even if they don't have the same operating system or platform as the safety expert. As the discussion evolves the architect starts to draft some corrections on the safety expert's drawing. Once both authors are finished working on the document in parallel mode, they are given the option of merging their modifications. This will be possible by replaying the events stored in the database on the other author's modified file. Upon conflicts among data modifications between author (person who started the application session) and users, the system will validate the author's

modifications only. If the conflict is among equal peers (neither author nor chairman) the modifications are prioritized according to their chronological order. The discussion continues and the contractor decides to jot down some of the ideas that are being generated. His avatar is then shown writing on the whiteboard in the 3-D environment. When the whiteboard is selected on the local desktop a 2D window appears to show more clearly what is being noted down. Once the session is over, the notes from the whiteboard are saved in XML format and the conference environment is again replaced by the social space.

3.2 Requirements

Much of the development of tools to support computer-mediated collaboration focus on the general needs of people while they are holding group meetings, for purposes such as: designing, taking a strategic decision, discussing a topic, finalizing a decision. Therefore it is not surprising that computer supported collaborative work tools are characterized according to two types of communication: synchronous and asynchronous.

Synchronous communication occurs when the participants are available at the same time, information is exchanged and extended in real time and feedback is available immediately. Examples of synchronous communication are groupware, audio and video conferencing.

Asynchronous communication occurs when the participants are available at different times, e.g., email, hyper mail, shared documents and World Wide Web browsing. Asynchronous communication has the disadvantage of delayed feedback and possible noise amplification. On the other hand, it has the advantage of leaving to the participants the time management of the information sharing.

Keeping in mind the synchronous and asynchronous modes of communication, the following set of requirements aim at enabling both types of communication as well as the following three models of behavior in collaboration as proposed by Sharples [1993]:

Parallel: all collaborators work simultaneously and send their work to each other or to an editor;

Sequential: the collaborators divide up the task into stages such that the output from one stage is handed on to the next collaborator in line;

Reciprocal: the group members work together to create the material, mutually adjusting their activities to fit the evolving output.

Based on the two types of communication (synchronous and asynchronous) and the three models of behavior in collaboration (parallel, sequential and reciprocal), the requirements can be divided into three main categories: communication media, information sharing and process management. This classification is related to previous work done in describing the requirements needed for CSCW systems. Saad and Maher [1995] proposed that CSCW tools must fulfill four high-level requirements:

Information sharing in which the representation of the design objects are shared using a language that can be understood by all the participants,

Communication media in which the collaborative design participants can communicate their intentions, planning and actions.

Process management where the participants can determine the stage of the process and what is to be done next,

Exploration space in which alternatives can be proposed, tested and changed.

Regarding the third group of requirements, the process management, previous work has determined that meetings require a control structure to facilitate interaction, which must be taken into account as part of the requirements of a collaborative system. Condon [1993] presents three models of control for the collaborative process in CSCW:

Fascist: one chairperson is always in control, having the right to decide who can use the keyboard at any one time.

Communist: the system is in control. Although it can only reflect the activities of other users, the system becomes the ultimate authority.

Anarchist: is a "free-for-all". Each user has the capacity of controlling and modifying the system.

Condon argues that the third model, anarchist, is "the only viable model for the design of CSCW systems." He also argues that "the main problem of groupware from the point of view of the system designer [is] the lack of a clear owner." However, the control structure model described in this thesis supports a different view in which, depending on the context, different controls are required.

Similar to Saad and Maher's division of the requirements into four parts (information sharing, communication media, process management and exploration space), Johannsen et al. (1996) argue that CSCW requires support in four areas: communication support, cooperation support, coordination support, and IT architectures.

3.2.1 Communication Support

Ubiquitous communications: It should be possible for distributed teams to communicate as efficiently as in co-located situations. To this end, a range of communication tools should be available for different work situations and locations (synchronous and asynchronous).

Audio/video Communications: collaborative work requires communication channels that are as "natural" as possible. For example, additional channels of non-verbal communication, such as facial expressions, gestures and body movements, complement spoken language. These aspects serve as additional cues in the communication process

and play a special role in a decision-making situation. Thus, high quality audio/video is needed.

Privacy and facilitating trust: Since people depend to a large extent on face-to-face interaction to develop trust among each other, in a physically distributed environment, information technology inherits the additional role of assisting the users in the building of trust.

3.2.2 Cooperation Support

Support of heterogeneous cooperation styles: collaboration may range from a very asynchronous style of remote cooperation to a highly interactive, meeting oriented style of synchronous cooperation. Therefore, it must be possible to tailor the interface to the actual needs of the distributed workgroup and to support the interaction between distributed workgroups using different cooperation styles. The cooperation style should not be fixed, as it should allow to be changed during a cooperation session.

Support of shared workspaces: cooperation requires the ability to share material or even more general, to have persistent shared workspaces. For shared applications, the protocols should allow to switch modes during a meeting. For example, cooperation may require a tightly coupled mode including "what you see is what I see" (WYSIWIS) when authors want to edit or discuss the same section of a CAD drawing or a loosely coupled mode when two authors work on different sections of a model but want to be informed of changes the other makes.

3.2.3 Coordination Support

Process coordination: An important function for the coordination of dispersed groups is the integration of synchronous and asynchronous group-work phases and to ensure continuous workflows though meeting control structures, provision of awareness of other people's activities and notification of important events. Regarding synchronous and asynchronous group work, most of the current collaboration solutions are primarily

designed for either "same time" support scenario or the "different time" scenario. This generally stands in contrast with the process orientation and the notion of continuous workflow in the office.

3.2.4 Architecture and IT-platform

Open systems architecture: Since members of a distributed team typically employ large, heterogeneous information technology infrastructure, the different software components of the member organizations must be interoperable. Due to rapid changes in these environments, it must be possible to extend and reconfigure the IS infrastructure on demand.

Application sharing: It is not sufficient for a distributed team "to be connected to each other on a common IT platform". Rather, different applications should be usable by all team members, where needed under a shared, cooperative interface.

After reviewing previous work done in the requirements for CSCW systems and looking at existing tools for collaboration, we found that there isn't a comprehensive tool that could address most of the requirements simultaneously. Current tools implemented for CSCW environments, either support communication (i.e. audio, video and voice over IP and limited data sharing), or support the work of a single-user, [Cicognani and Maher 1997]. A third group of existing tools are those solutions for organizations that work with smaller or less complicated projects such as FTP sites or project-specific Web sites. Drawing from the review of the current solutions, the confluence of tools for communication, multi user application session and project repository combined with meeting control structures will begin to provide an environment in which CSCW can be enabled and studied.

3.2.5 Group Dynamics-Aware Conferencing

Previous work in design team interaction and group dynamics provide a greater understanding of group interaction modes. It is obvious that certain elements of physical interaction cannot be replicated with simple audio and video communication. To facilitate the flow of conversation in group discussions, the elements of engagement and attention are critical. This section discusses the mechanisms implemented in CAIRO to support the Process Management requirement in an engineering problem-solving setting. They are

A Sense Of Place: The notion of place where members meet and share persistent objects is of overbearing importance when we talk about a discussion. Therefore, it is critical to include mechanisms in the system and its user interface that clearly portray entrance and egress of individuals as well as their relative stance with respect to others in the meeting.

Spatial Interaction: A groupware conferencing tool must support deictic referencing in both gaze and pointing. Hence, the tool must have a pointing feature that clearly distinguishes between hearers of conversation and those to whom the conversation is addressed. Since groupware tools usually have large set of interaction tools it can be distracting and so it is necessary that the focal tool should be distinctly identifiable.

Degrees of Engagement: The tool should have flexibility in the design so that the participant can have greater control in deciding his/her intent vis-à-vis degree of engagement, addressed in the requirements summary as item (i) Parallel, Sequential or Reciprocal collaboration behavior. Also, pending speaker queue should be prioritized in order to allow for urgent commentary in an online meeting and the queue should allow simple disengagement from the conversation. Finally, the status of every participant should be visible to all participants in the meeting.

Floor Control Strategy: The notion of floor control strategy (e.g., chairman controlled, brainstorming, and lecture) is an important requirement listed as Process Management. In regular meetings a floor strategy is usually adopted either explicitly or implicitly. These strategies govern floor control and define a particular meeting style. Effective choice of floor control strategy can improve the quality of the collaboration effort. Also, a more complex floor control strategy is an option that can be exercised from review of the interaction inputs provided [Sen 1999].

Drawing from this revision of previous work in requirements, an outline of the functionality of the system has been made. The different types of communication (synchronous and asynchronous), the models of behavior (parallel, sequential and reciprocal), and the main categories of requirements (data sharing, communication and process management) determine that the integrated system should posses the following functionality:

(i) Parallel, Sequential or Reciprocal collaboration behavior: Ability to support synchronous (parallel collaboration behavior) and asynchronous modes of collaboration (sequential or reciprocal collaboration behavior) as well as the transition between the two.

(ii) Hierarchically structured groups: ability to support groups in the various stages of formation, i.e. the ability to have hierarchically structured groups that are easily expandable.

(iii) Information Sharing: synchronous visualization and underlying object manipulation as well as the transition to asynchronous shared workspaces. For example: users working on the same file in parallel mode should be given the option of merging their modifications.

(iv) Process Management: activity awareness, event notification and conference control mechanism is required to provide efficient group interaction. The system must be adaptable to different conference styles, from informal, unstructured

conversation (brainstorming) to a stringent and formal conversation control mechanism (chaired meeting).

(v) Communication Media: multiple media channels are required for group communication, generally audio, textual, and visual data.

(vi) Social Feedback and Trust Facilitating: to provide nonverbal behavior information and privacy in order to foster trust and effective communications.

(vii) Open Architecture: since design teams often use different software components on heterogeneous platforms and operating systems, it is important that they are interoperable to allow collaboration and data sharing.

(viii) Group memory: ability to retain group memory to build corporate experience as specified by the adjourning phase in the group life cycle.

(ix) Sense Of Place: The notion of place where members meet and share persistent objects.

(x) Spatial Interaction: A groupware conferencing tool must support deictic referencing in both gaze and pointing.

3.3 Summary

Different subsets of the requirements described in this section have already been addressed by applications developed at MIT. These applications will be described in more detail in Chapter 6. These requirements and an overview of the existing applications will then lead to a design of a broad band Collaborative ASP shared workspace described in Chapter 7. The following chapter will introduce the concepts behind generic shared workspaces for CSCW and their applications.

Chapter 4

4 Shared Workspaces for CSCW

A shared workspace for CSCW can be defined as the remote sharing of computer display surfaces between people involved in common tasks and who collaborate without leaving their regular workplace [Fluckiger 1995]. However a more general definition could be: the computer mediated ability to allow distributed people to jointly refer, display, and manipulate objects of common concern [Schefström 1999]. The purpose of a Shared Workspace is the remote sharing of the "work entities" of concern to the task at hand, such as programs, drawings and documents. For professional applications, such as distributed architectural design projects, the ability to share work objects is more important than audio and video [Maher, 2000].

4.1 Shared Applications

A shared application, sometimes called "Tele-operation", is the ability to not only share an application, but to be able to manipulate the underlying objects. A shared application must allow two or more users to view and manipulate the same document or data in what is usually called "What I See Is What You See" (WISIWYS).

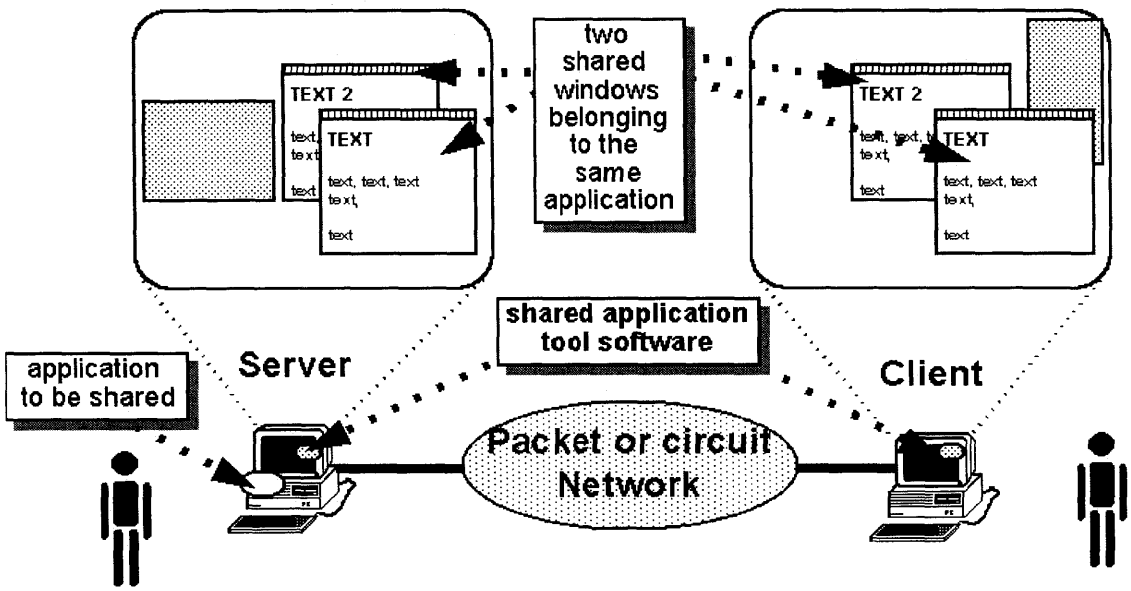

Figure 4-2. Bipartite Use of Shared Application Tools [Fluckiger 1995]

In the case of shared applications, it is not only the sharing of the screen pixels, but the underlying program and operations. The two implementation methods used in this research were (a) the remote duplication of the input and output behavior of a tool as well as (b) display remotely the presentation layer (GUI) of a tool. In this way, multiple users can synchronously discuss while referring to the same executing program or stored document. Input requires rather tight "floor control" to avoid chaos, as shown by an experiment during this research when seven participants tried to work on a CAD drawing simultaneously.

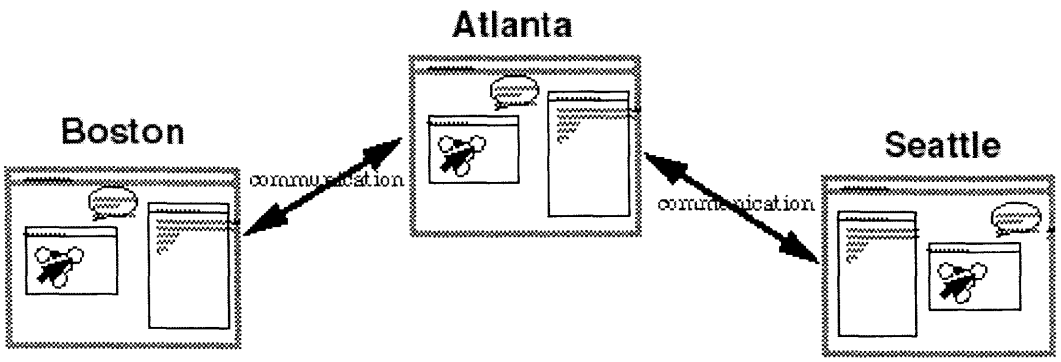

Figure 4-3. Shared Application [Schefström 1999]

4.1.1 Shared Whiteboard

The classic example of a shared workspace for CSCW would be the shared whiteboard. A shared whiteboard allows distributed users to communicate by drawing graphics on a shared canvas, either a white background or an "imported" document. The joint viewing is obtained by the distribution of pixels to all the collaborators. Therefore there is little involvement with the underlying "semantics".

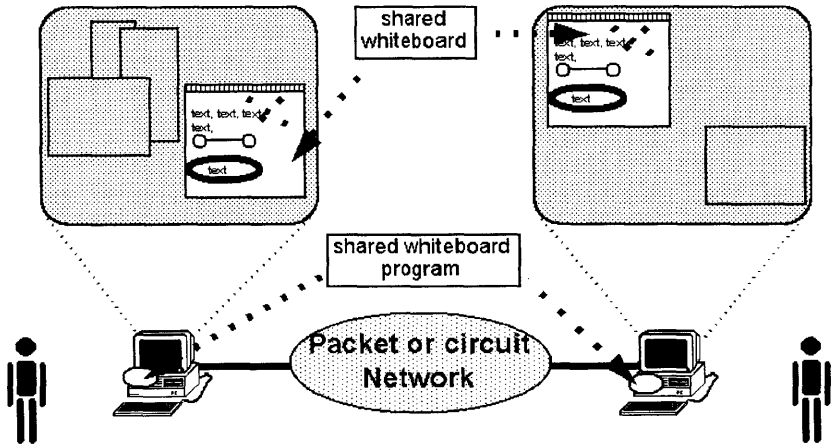

Figure 4-1. Bipartite Use of Shared Whiteboards [Fluckiger 1995]

These shared whiteboards have been purposely built for collaboration, which in principle allows for high quality collaboration support, as they can be made "Collaboration aware".

The way these whiteboards work is by creating one instance of the shared whiteboard program at each participant's computer, distributing commands to the other instances. Usually these applications have no Floor Control policy, so each participant can input to the whiteboard without restrictions creating certain confusion, especially when there are more than two people logged in. This input mode is called "implicit locking", since locking takes place automatically as a user takes action. In these cases granularity (ownership over parts of a shared artifact) is an issue.

Chapter 5

5 Methodology

The development of different strategies for the integration of a shared workspace into the framework of a generic collaboration application consisted of two phases. The first phase was the "High Level" Solution or Multi-User Extension. The second step was to take the knowledge gained from the previous phase and apply it to the actual integration of a shared workspace with a meeting environment using a "General" Solution or Shared X. Both of these procedures are detailed in the following sections.

5.1 "High Level" Solution

A good alternative to building applications that directly support distributed and shared usage is to add a multi-user extension to existing applications. The principle on which this is strategy is based is: "If all involved applications receive the same input and data in the same order, they will present the same behavior to all users". Therefore in order to convert applications from single-use to support multi-user shared usage, an extension is added to the original application that can access the functionality of the original application, so by catching input from each user and sending the information to the other users, the output is recreated on the other users computers. One of the advantages of this solution is that it can be less complex, and allow for tuning to the circumstances, both with respect to user interface and communication characteristics. It has a great potential, since it also allows for collaboration awareness.

5.1.1 Multi-user Extension

The objective of the multi-user extensions that were developed was to add distributed and shared usage to existing application that had been designed for single users. As explained previously, the principle on which the extensions are based is very simple, by giving all the participants the same input; the output for all will be equal. Each distributed application is listening for a specific set of events and once these events are triggered, they are sent over the network to the other distributed applications, thus obtaining synchronization. In other words, by catching input from each user and sending the information to the others, the output on the other users computer is recreated (Figure 5-1). The following figure shows basic components of the multi-user extension with a simple case of three users.

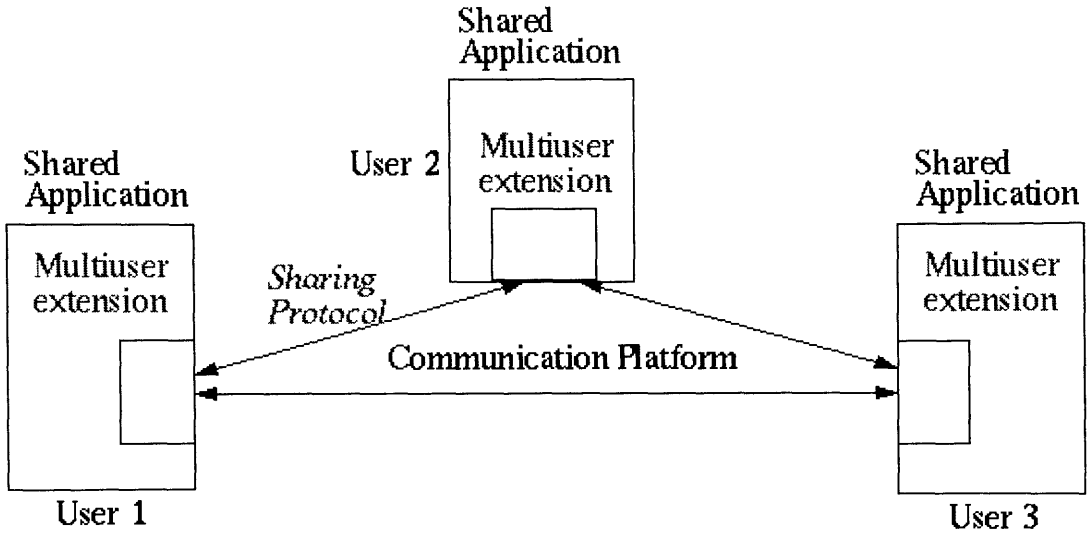

Figure 5-1. Multi-User Extension Architecture [Schefström 1999]

For the prototype developed, simple proprietary message passing was used as the Sharing Protocol. So each application for which the extension was built had a set of messages that were mapped to the each event that was generated by the application. A possibility is that future applications might include a Sharing Protocol as a standard component. This protocol could be based on the T.120 standard defined by the International Telecommunications Union [ITU 2000].

The applications that were targeted for the prototype developed of this research were the office applications (Word, PowerPoint and Excel in Figure 5-2), a web browser and a CAD tool. The extensions were developed using Java and consisted of two main modules: the networking module and the application access module.

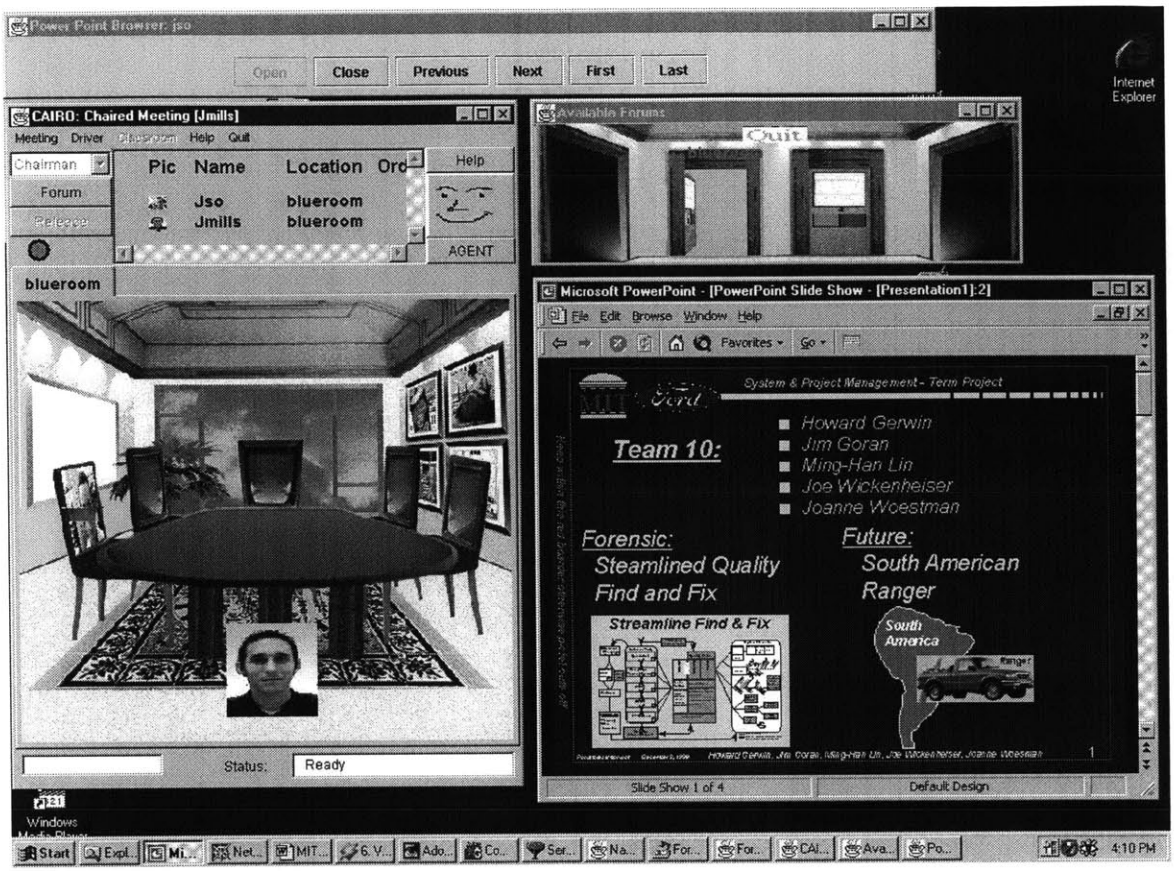

Figure 5-2. Synchronized Presentation Using Microsoft PowerPoint

The networking module's main responsibility is to listen to events that are being sent from the other distributed applications as well as sending events that occur on the local application. The application-access module is in charge of accessing the functionality provided by the application. Each targeted application had an associated COM object that provided an interface to the functionality of the application. By using a bi-directional pure Java-COM bridge [Jintegra 2000] it is possible to make calls on the internal functions of each application. J-Integra's pure Java runtime talks to COM components using Distributed COM (DCOM) layered over Remote Procedure Calls (RPC), which are themselves layered on TCP/IP. So at the lowest level J-Integra uses the totally standard Java networking classes. Figure 5-3 explains the components of the application-access module.

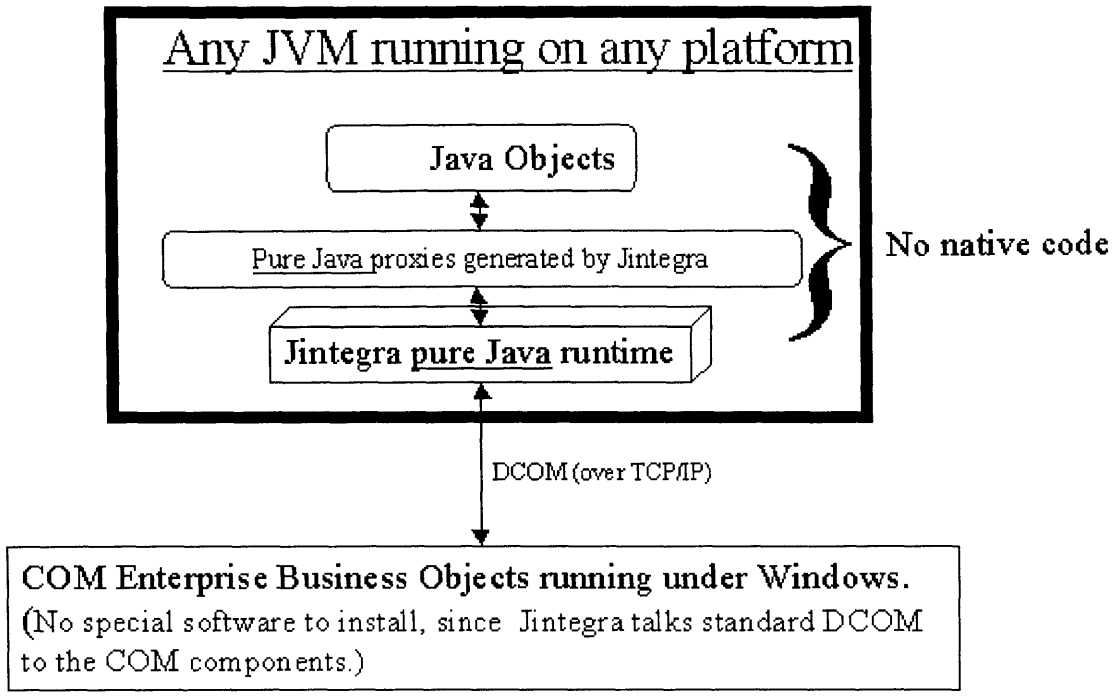

Figure 5-3. Application-access Module [Linar Ltd 1999]

One of the advantages of this solution is that it can be less complex to implement, and because a multi-user extension must be developed for each application, it allows for some customizing according to the circumstances. For example the user interface as well as communication characteristics can be designed for each specific application. In this sense it has a great potential, since the application is in a certain way is 'aware' that it is used in a distributed fashion. And new collaboration-specific functionality can be added to match the needs of the distributed team that is using the application. This is an important difference with the Shared X solution explained in the next few sections, where the application is deployed 'as is' with no possibility of modifying its functionality.

5.2 "General" Solution

At the other end of the spectrum of possible implementation techniques we find a "general" solution, based on the X Window System. This is because X Window has an

inherent element of distribution, the following image outlines the basic concepts and illustrates the distributed nature of the X Window System.

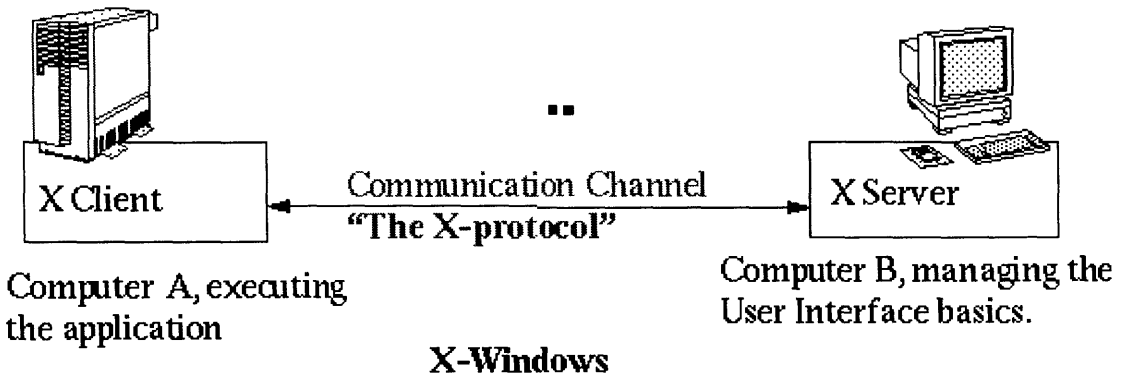

Figure 5-4. X Window System [Schefström 1999]

A natural idea is then to allow applications to be shared by dropping or inserting information on the X protocol channel, and distribute the information to more than one X Server. The advantage of this solution is that all programs developed for X Window can now, in principle, be shared in a multiple user context. The disadvantage of the solution is that it seems to be rather complex to make it work in practice.

5.2.1 Shared X

This approach to the problem of adding distributed and shared usage to current application by using shared X is a 'general' solution in the sense that there is no application-specific software modules to implement. This solution is based on the X Window System because its design is already inherently distributed. The presentation layer (which includes the user interface) is separated by design from the applications logic, allowing for remote displaying of the user interface. So the way to implement shared applications is inserting information on the X protocol channel and distributing the information to more than one X Server. Figure 5-5 explains the components needed with

a case of three synchronized X servers and one X client. The only software module that needs to be implemented is the **Shared X Switch** that acts as middleware in this 3-tier architecture. The switch is in charge of receiving X protocol from the X client and sending it to the X servers that want to join that particular application sharing session.

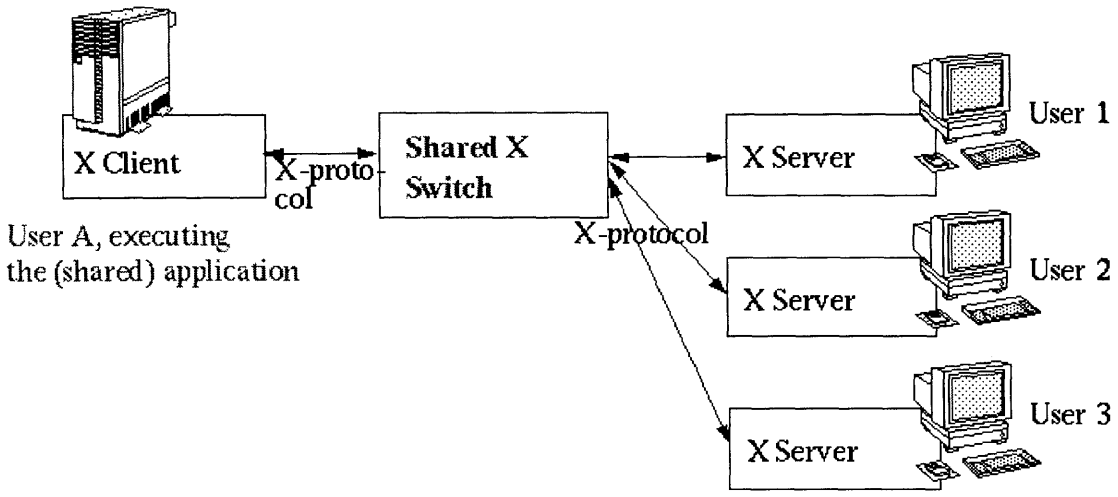

Figure 5-5. Shared X Architecture [Schefström 1999]

5.2.1.1 Shared X Switch: XMX

A shared X switch was not developed for this research, however previous work in the area allowed us to experiment with an implementation of a shared X switch. We tested XMX, an X protocol multiplexor, implemented by John Bazik at Brown University. XMX provides a WYSIWIS (What You See Is What I See) environment; it paints the same graphics on all participating displays as explained in the previous section. The shared client applications appear to each participant in a virtual root window that is subject to local window management. In this way, the shared X session coexists with each user's private X session. The X client applications that are shared via XMX are unaware that they are being viewed or controlled by more than one user.

Existing, single-user X client applications may be shared using XMX without recompilation, re-linking or access to source code. XMX is designed as an application-

sharing server. It provides both the switch mechanism for shared X and a way to control the shared session from other programs. Like X, it leaves policy decisions to application programs. In this way, XMX may be used as a component in any of a variety of systems that require application sharing. Because XMX is interposed between X clients and X servers, it cannot jump into an existing client-server session and begin sharing it. XMX can only share X sessions it was involved in from the start [Jones, 1993].

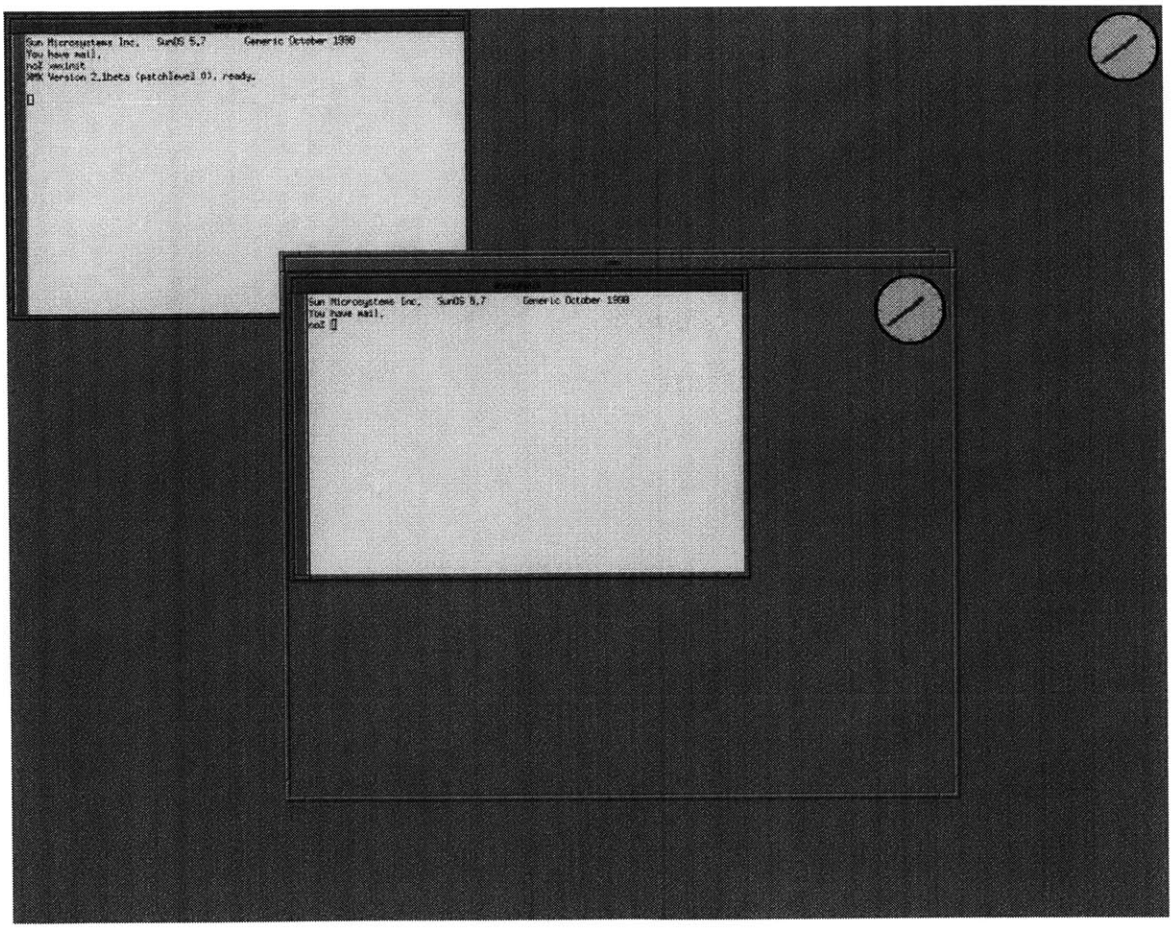

Figure 5-6. XMX Virtual Root Window, a root-window-in-a-window

When sharing an X session like this, it is necessary to have participants (X servers) join and leave the session and to control which participant(s) can interact with the X

clients while the others watch. In this sense, XMX includes the X Multiplexor Control (XMC) client application where the X Protocol Switch functions can be visualized. Figure 5-7 shows the user interface of XMC where the people icons represent participants in the session. The figure shows two virtual root windows on a machine named "no". The buttons named Floor, Seat and View allow to control which displays may provide keyboard and mouse input to shared applications.

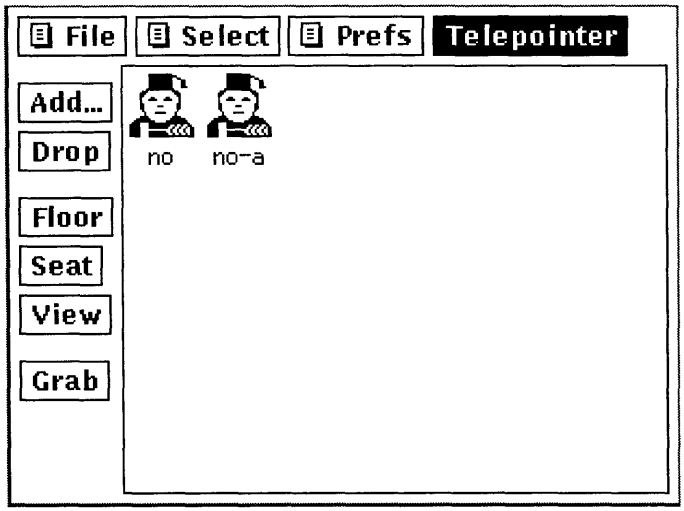

Figure 5-7. X Multiplexor Control (XMC) User Interface [Bazik 1999]

XMX provides three input modes for participants. These input modes dictate what happens when a user types or moves the mouse in a virtual root window. They are named for the roles one might have in a formal meeting. When a participant has the Floor, the keyboard and mouse inputs control the shared applications in the virtual X session.

View mode is the exact opposite of floor mode. When a display is in view mode, the user may only watch passively what happens in the session. All mouse and keyboard input is ignored.

Seat mode is between floor and view modes. When a display is in seat mode, the user's mouse and keyboard input is by default ignored by the X clients in the shared session, just like in view mode. But an XMC client can choose to see input events from

displays in seat mode. Seat mode gives the participant the right to be recognized and participate, even though the participant is not running the session at the moment [Bazik, 1999].

The most visible advantage of this solution is that all programs developed for X Window can be shared in a multiple user context without recompilation, re-linking or access to source code, which in most cases is proprietary. However, there are several disadvantages to this solution. The first one is that the implementation is rather complex to make it work in practice. The second drawback is that since the idea builds on a single central instance executing in a single X client, with displays at multiple places, scalability is not obvious. The third disadvantage is that the coupling between users is very tight. Users cannot make private "deviations", (such as temporarily looking at other objects or documents). So this solution is strictly 'I See What You See', while the multi-user extension allowed certain flexibility among participants. The fourth drawback is that unlike the multi-user extension, this solution is not collaboration aware. The underlying application does not "know" that it is shared, so it cannot by definition provide specific collaboration support.

5.2.1.2 SCO Tarantella

However the most important disadvantage with Shared X solutions is that they can only be used with applications that run on the X Window System, namely Unix. Therefore none of the Microsoft applications can be made collaborative with this approach and none of the clients can be devices other than X servers. So there is a server hosting and a client access problem that would need to be solved if this approach were adopted. The client access problem was solved during the testing phase by using SCO's Tarantella X emulator applet. By using Tarantella, users can access XMX from any Java™ technology enabled client device, such as web browsers, without the need for additional software to be installed. Tarantella uses a three-tier architecture that integrates the application server that's running XMX with diverse types of clients. XMX continued

to run on our initial Unix server, and now different client devices were able to access XMX, such as PCs, UNIX® workstations, and Network Computers.

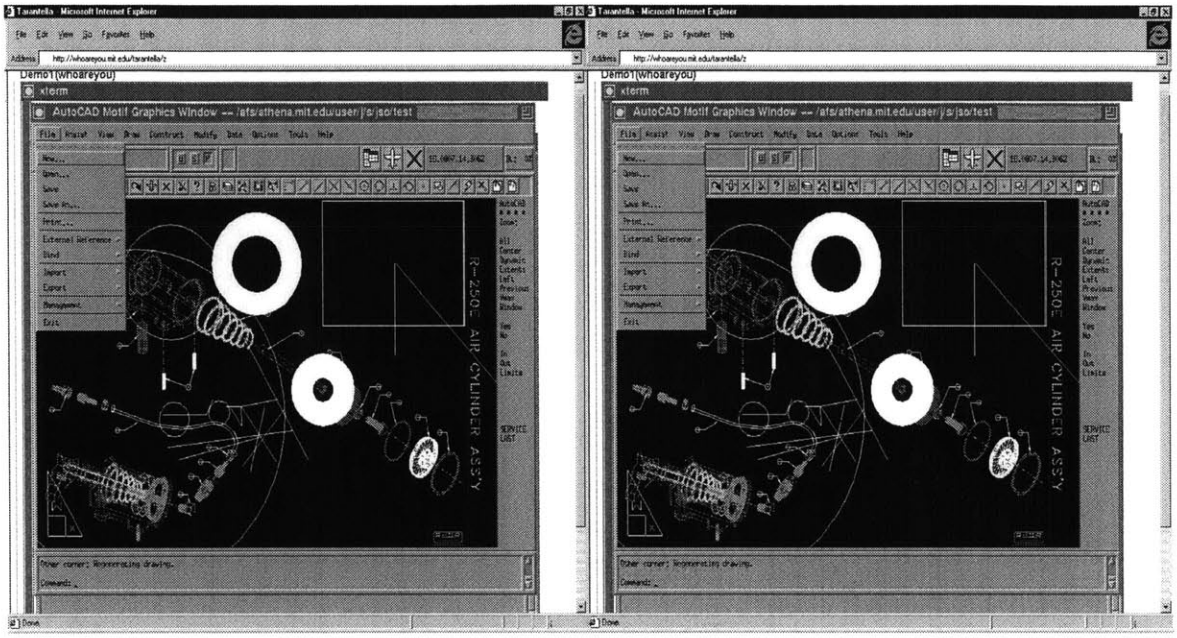

Figure 5-8. SCO's Tarantella X Emulator Applet Running XMX

5.2.1.3 JCraft WiredX

SCO's Tarantella product is not the only X emulator applet that is available on the market. Another free software solution that was tested was JCraft's WiredX applet emulator, JCraft is dedicated to the Java platform, and one of its products is a fairly advanced X emulator applet. A screen shot of the results is shown in Figure 5-9.

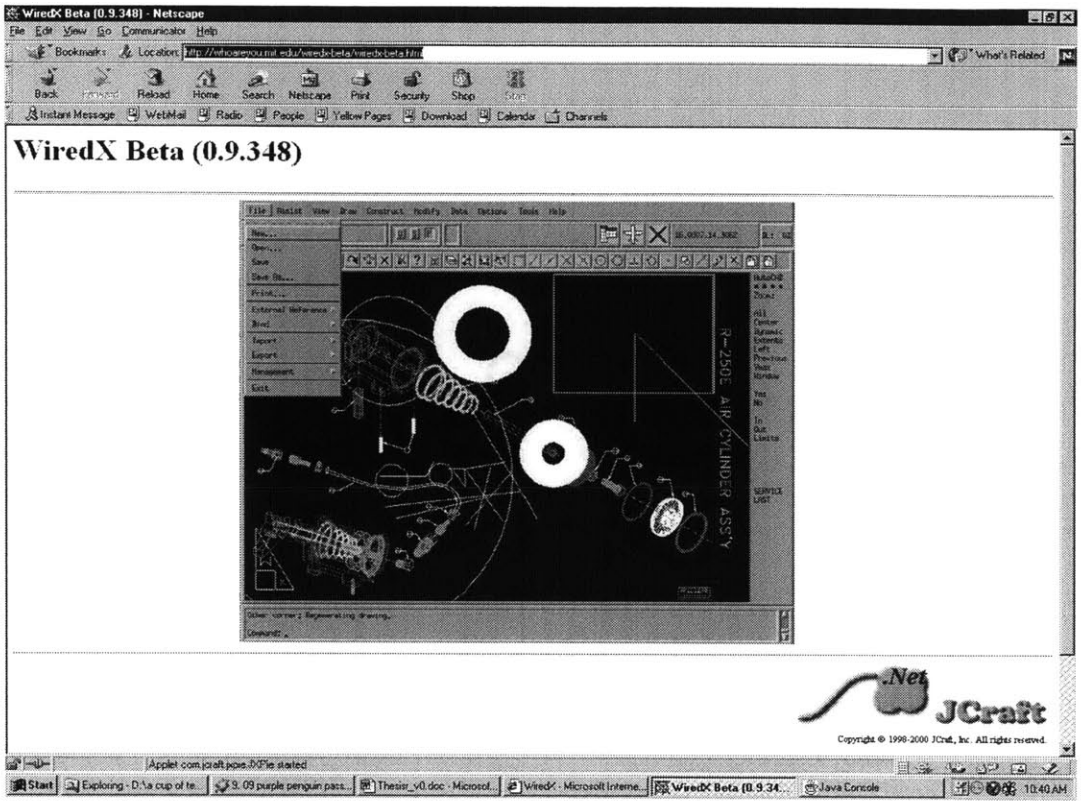

Figure 5-9. JCraft WiredX X Emulator Applet Running XMX

The results of using JCraft's WiredX X emulator applet are very similar to SCO's Tarantella product. The underlying concepts and functionality are similar. So, other than some performance differences and initial color mapping difficulties, the results were comparable to SCO's

5.2 Summary

After giving an overview of the two main strategies pursued to create a shared workspace, a description of CAIRO developed at Massachusetts Institute of Technology and VNC developed at the Olivetti and Oracle Research Laboratory will be given. The products of the development accomplished so far at MIT and ORL have addressed some of the issues raised in the requirements of this paper. So, a better knowledge of the functionality of these two separate applications will help understand the advantages of their integration into a single system.

Chapter 6

6 Integration Perspectives

As stated in Chapter 1, the focus of this research is to develop a shared workspace prototype, based on the ASP delivery model of a collaborative design system. This system integrates collaborative applications that have been developed at ORL with an existing collaboration system, CAIRO, developed at MIT. However, to even begin considering an integration of two separate systems, one must first understand the underlying concepts behind each of the tools. A deep understanding of the issues that had to be considered in building the final product is also required. In this chapter, overviews of the CAIRO collaboration system built at MIT and the VNC applications built at the Olivetti and Oracle Research Laboratory are provided.

6.1 The CAIRO System

The Collaborative Agent Interaction control and synchROnization (CAIRO) system is a distributed meeting environment that was developed at the Massachusetts Institute of Technology specifically for AEC project development [Benjamin 1998]. This system was created as part of the Da Vinci initiative whose goal is to "explore computer support mechanisms for enhancing distributed engineering design change negotiation" [Pena-Mora, et. al. 1996]. The CAIRO effort develops a methodology for computer-supported coordination of distributed multi-disciplinary design meetings using the Internet. Conferencing is often one of the more difficult aspects of large-scale projects, especially those involving participants that are not collocated. Therefore, one of the primary goals of the CAIRO endeavor is to add structure to the collaboration process in the virtual space by defining various protocols of interaction that are commonplace in the physical space, as specified by the Process Management requirement in Section 3. The functions of the virtual environment typically imitate the functions of the real world [Benjamin 1998].

The CAIRO initiative has taken the meeting protocols mentioned above and implemented them in its distributed meeting environment. The shared workspace for this collaboration scheme is based on the functionality provided by a whiteboard. The original CAIRO system contains no mechanism to support a more comprehensive shared workspace. The new collaboration system is designed incorporate a new shared workspace that will include other applications as well as multiple platforms to enable collaboration among members on a heterogeneous network.

This research looks at how to apply the evolving Application Service Provider delivery model into the CAIRO collaboration architecture. This new delivery model, using an underlying server-side computing architecture, complements CAIRO's original whiteboard and image sharing to create a new Broadband Collaborative Application Service Provider model. Moreover, CAIRO will assume the role of a meeting facilitator using an information policy that defines and controls the flow of information among the

thin clients. This information policy not only defines the protocols for the meeting but also the availability of information to participants.

6.1.1 The Architecture

The distributed architecture of the CAIRO system is built upon the concept of client/server technology. Figure 2-1 illustrates an instance of the CAIRO system architecture. In this representation, there are four participants (a, b, c, and d) in the discourse. The design consists of three main elements: the **Collaboration Manager**, the **Forum Server**, and the **Name Server**. The following sections explain the roles of these constituents.

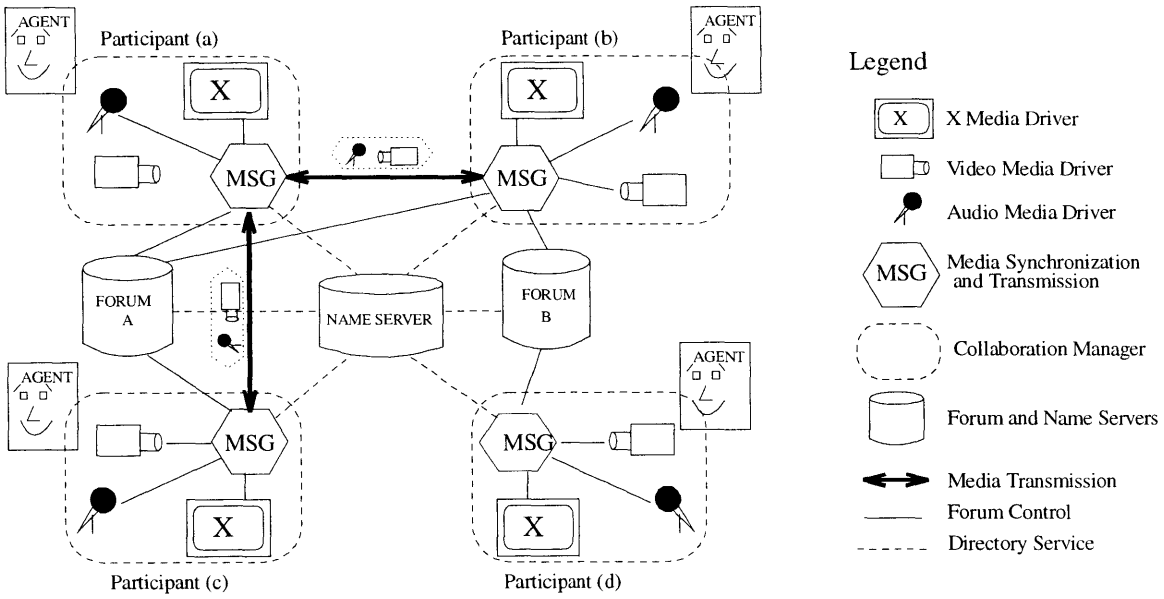

Figure 6-1. Current CAIRO Architecture [Hussein 1995]

6.1.1.1 Collaboration Manager

The collaboration manager is comprised of a number of media drivers and a message server. It also possesses a Graphical User Interface (GUI) that allows the user to easily navigate the system (Figure 6-2). Encapsulated within the interface are numerous

other tools, menus, and visual analogies that facilitate collaboration among participants. The interface and its features are discussed in Section 6.1.2.1.

6.1.1.1.1 Media Drivers

The term "media driver" describes a set of applications that users employ to transfer various forms of data to other meeting members. The original version of CAIRO consisted of a set of drivers that were specifically geared towards engineering project development. These drivers, which can also be seen in Figure 6-2, include a text driver, a whiteboard, a scheduler, and a design rationale and implementation tool (not pictured). The text driver permits the exchange of text messages among the clients so that they can communicate through the use of written language. The whiteboard provided an initial shared workspace that simulates an actual physical whiteboard that may be found in an office setting. This driver allows members to communicate information in the form of sketches or drawings. It is also capable of loading images and transmitting them to other participants. The scheduling tool includes an interface that describes a plan in the context of time. It lets users create and modify a schedule in a group setting. Lastly, the design rationale and implementation tool enables participants to collaborate about design issues of an engineering project. The availability of these tools greatly augments the ability to convey specific engineering ideas from one person to another.

6.1.1.1.2 Message Server

This server handles all transmission of information between users and keeps track of membership in forums. The term "forum" is defined in Section 2.1.1.2. Messages that are exchanged between various participants in a forum are all in the form of TCP/IP datagrams. Each participant has his/her own handler that interprets those incoming messages and routes them to the appropriate drivers.

6.1.1.2 Forum Server

A forum is defined as "a structured group of participants involved in a collaborative effort" [Hussein 1998]. In this document, the terms "forum", "meeting", "engagement", "conference", "session", and "collaboration instance" are used interchangeably, and all refer to this same definition. The forum server forms an analogy to the actual, physical meeting space. It keeps track of all of the individuals involved in a specific meeting and their current state (see Figure 2-3). A member of a forum may be in one of three states: active (logged in and listening), speaking (has control of the floor), or non-active (not logged in). In addition, the Forum Server maintains a log of events that occur within the forum for playback purposes. Thus, a user can go to specific entries on the agenda and replay the events that took place during that agenda item. The playback of the recorded log is only seen by the individual who requested the replay and is not saved in the event log.

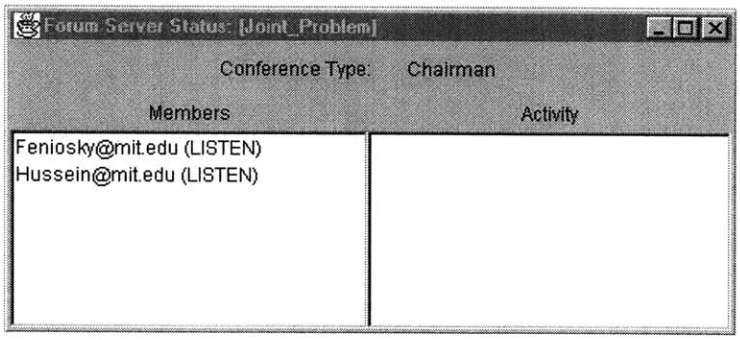

Figure 6-2. Forum Server Interface

The forum server also has the responsibility of regulating floor control, one of the most important features in CAIRO. Three different meeting styles are presently defined: chairman, freestyle, and lecture. Each of these styles exercises a set of controls to regulate when and to whom users can speak.

6.1.1.2.1 Chairman Meeting

The chairman meeting style within CAIRO is modeled after the real world chairman meeting. In this meeting strategy, one participant is designated by some prior decision to be the chairman of the engagement. This person is then given authority to determine who gets control of the floor. The chairman himself can take control of the floor as he/she wishes. However, other members of the meeting must request permission from the chair to speak. The chair then decides whether he/she will grant permission to the requestor. These actions are performed using the menus described in Section 2.1.2.3.

6.1.1.2.2 Freestyle Meeting

The freestyle meeting style has an informal composition. Unlike the chairman style, no one is in charge of the meeting. Instead, members are allowed to take the floor and speak of their own accord. This style of meeting is common for brainstorming sessions in order to keep the conference proceeding smoothly. People can speak freely as ideas flow into their minds, and thus, can contribute ideas without the constraint of obtaining permission from another individual.

6.1.1.2.3 Lecture Meeting

The lecture meeting style is, as can be inferred by its name, analogous to a lecture or classroom setting. In this case, the lecturer assumes a role similar to the chairman in the chairman strategy. However, primary communication is directed to the lecturer and no communication takes place between participants in the lecture.

6.1.1.3 Name Server

The Name Server centrally manages a global directory for the CAIRO system. This server catalogs a variety of information that can be retrieved by any user. This information includes the name, location, and status of each participant. Likewise, it retains knowledge about the forum's name, location, and status. When a user logs into the CAIRO system, the Name Server can provide information to the user such as the other

individuals who are logged into the system and what forums are available to the user. Unbeknownst to the user, information about the Internet address of the other participants and the forums are sent to the client as well. Figure 2-4 exhibits the name server interface where the list of forums and users online are displayed.

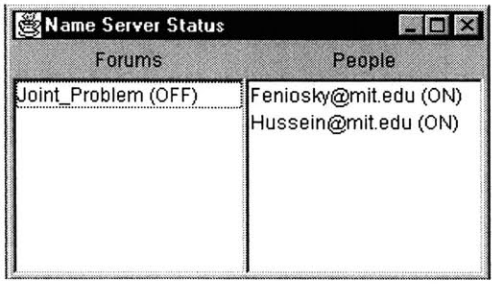

Figure 6-3. Name Server Interface

6.1.2 The Features

The following sections provide a brief description of the features included in the present version of CAIRO.

6.1.2.1 The Interface

The screenshot shown in Figure 6-4 provides an idea of the main CAIRO interface. There are several components of the interface that require an explanation and how they fulfill some of the requirements discussed in Chapter 3:

Meeting Control: provides meeting control information such as who has permission to speak, who has control of the floor, who is currently speaking and who has requested permission to speak and who is requesting a side-talk. A conference control mechanism is one of the key requirements of this distributed tool. The requirements section elaborates on the necessity of conference control and it is one of CAIRO's key precepts as well. Thus, there are rigid guidelines that embody and enforce the particular type of conference control needed.

Collaboration Manager: 2D interface that provides a visual metaphor of the collaboration session. The participants are represented by click able images gathered around a table that allow initiating communication. This component of the interface fulfills two requirements that were considered key during the design of CAIRO: to create a *sense of place* and simulate *spatial interaction.*

Video and Audio Interface: facilitates exchange of synchronized audio/video data between multiple collaborating team members over the Internet as formulated in requirement (v) Communication Media. Multimedia channel synchronization is essential due to random delays inherent in the underlying network. It should be recalled here that video communication without the audio to go with it (lip-sync) can be very irritating and anything that will make the distributed interaction process less effective is really not desired. Thus, CAIRO has a system by which the receiver reassembles the multimedia frame and ensures that playback of the frame is synchronized so that it reflects the initial input form the source.

Shared Whiteboard: that simulates an office environment where meeting participants can communicate visual information, such as simple sketches or images of various product design ideas. The shared whiteboard belongs to the group of applications that fulfill the Information Sharing requirement.

Text Driver: A text chat box that allows sending written messages. This component belongs to the Communication Media category of the requirements, along with audio and video, and is seen as an alternative tool for low bandwidth users.

Agenda Tool: In partial fulfillment of the Process Management requirement, the agenda tool contains an outline of topics to be discussed in a meeting. The agenda in CAIRO works in conjunction with the forum to control the flow of a meeting, for example by recording the duration of each agenda item.

Database Support: as part of the shared workspace, collaborators can be assured their transaction is seen by all other collaborators since the database alerts all

collaborators of the change (process management requirement). Also by utilizing a central repository for data collection and event distribution, many interfaces can be developed from a single data source allowing WAP enabled phones to interact with Web browser clients (ubiquitous communications requirement).

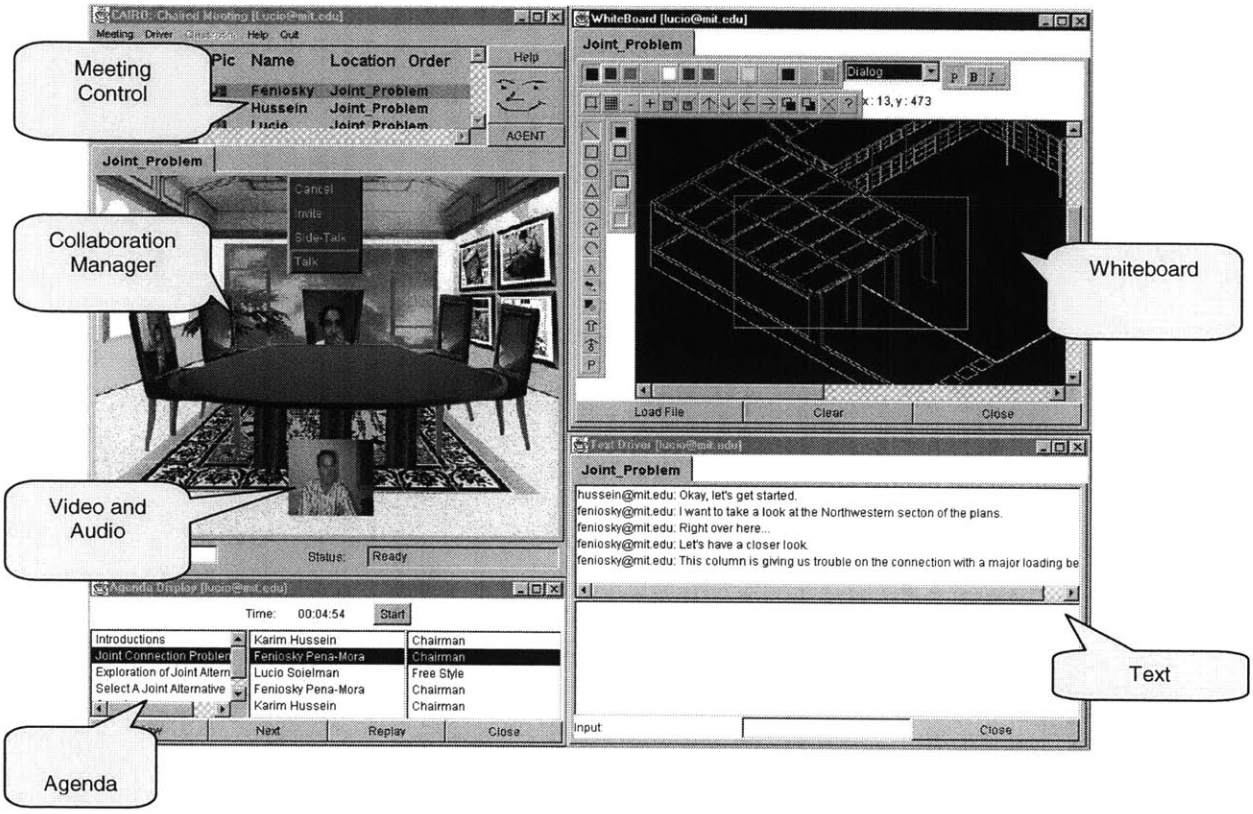

Figure 6-4. CAIRO Interface

One of the intended advantages of the CAIRO meeting environment to provide a number of visual analogies to the actual physical meeting spaces. As an example, the image of the table visible in the Collaboration Manager in Figure 6-4 can take on two different shapes. The round version cues the viewer that the meeting he/she is participating in is a freestyle meeting. In contrast, a rectangular shaped table denotes a meeting with a chairman. These visual analogies have dual concepts in the physical world that may be familiar to the typical individuals.

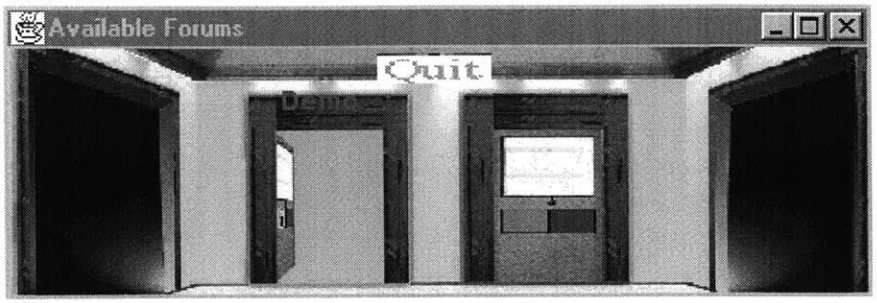

Figure 6-5. Hallway of Meetings

Figure 6-5 another instance of a visual analogy. In this window, the analogy of a hallway is used to represent the choice of meetings that are available at a particular instance in time. Each door represents a separate meeting and has a title describing the meeting over it. At the left and right sides of the image lie extensions to the hallway through which users can to reach more rooms. Once a user finds the room that he/she would like to join, he/she simply clicks on the corresponding door. It then opens as shown in Figure 6-5. An open door signifies that a room has been entered. Therefore, this doorway represents an entry to our virtual meeting. This feature tries to provide a sense of place and spatial interaction as discussed in the requirements chapter. The following paragraphs describe the most relevant functionality of CAIRO and how it satisfies the requirements discussed in Chapter 3.

6.1.2.2 The Agenda Tool

As in a physical meeting, the agenda contains an outline of topics to be discussed. The agenda in CAIRO works in conjunction with the forum to control the flow of a meeting. The agenda recognizes the duration of each agenda item. The agenda tool, presented in Figure 6-6, is simply comprised of a window that has a timer to keep track of the time spent on an agenda item and a view of the items in the agenda. The panels, from left to right, identify the name of the agenda item, the person in charge, and the style of meeting.

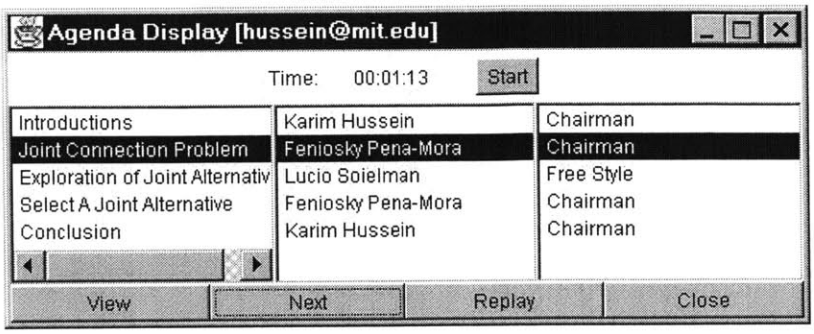

Figure 6-6. The Agenda Tool

6.1.2.4 Side-Talk

Seeing the need to support hierarchically structured groups (requirements ii), this feature allows a subgroup of a conference to speak privately during a meeting. One can conceive a scenario in which a subcommittee is formed to discuss a side issue. In this case, the side-talk function could be used for the purpose of the subcommittee. Lastly, participants of the Side-Talk can switch between tabs, and by doing so; can alternate their participation between the main meeting and multiple Side-Talk instances simultaneously.

6.1.2.3 The Agent

The agent is another tool created as an answer to the Process Management requirement. The agent is a program that monitors the actions of the users, residing with the moderator of the meeting. The agent procures information from the interactions among users and tries to make intelligent suggestions to the moderator that may increase efficiency in the meeting proceedings. Currently, the CAIRO agent possesses only limited functionality, focusing on the agenda, certain textual cues, and request activity.

For example, during a meeting, if the time allotted for an agenda item has expired, the agent will prompt the moderator to continue to the next agenda item. In this case, the agent is a passive timer for the user. The client can choose to accept the suggestion or reject it depending on how he/she feels the meeting is proceeding. However, the agent

still serves as a reminder that keeps the participants on topic and in-line with their meeting agenda.

The agent also reacts to certain textual phrases that are sent and received through the text driver. It parses the text that is being delivered for specific key words. For example, if the agent sees a word such as "explain", it may suggest to the moderator that he/she change the meeting style to a lecture control strategy. If the word "discuss" were used, then the agent would suggest that the freestyle meeting strategy be employed. Again, the user can then decide whether to accept the suggestion or not.

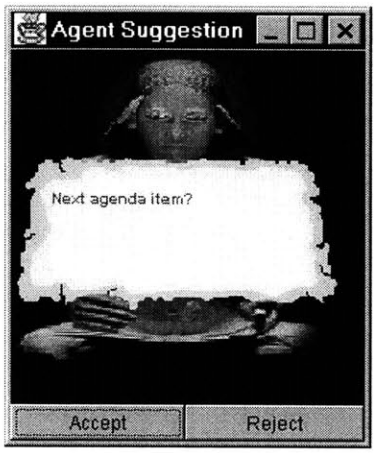

Figure 6-7. The CAIRO Agent

Finally, the agent also responds to actions that are being taken by the users. If the agent notices that a lot of requests to speak are being made by the participants, then it may recommend that a freestyle meeting strategy be applied. This way, the meeting may run more smoothly if users do not have to constantly request permission to speak. Figure 6-7 shows a sample view of the CAIRO agent.

6.1.2.4 The Social Agent

Effective communication involves many factors that include the verbal as well as nonverbal aspects. In regard to the nonverbal factors associated with communication and social interaction among people, gestures undoubtedly convey and express many subtle

as well as obvious meanings. As an initial response to the Social Feedback requirement, the Social Agent was designed and developed as part of CAIRO [Su 1998]. The main features that were targeted were **Expressions**, **Casual Contact, Awareness Driver** and **Three Dimensional Virtual Environment**. These features are explained in the following sections.

6.1.2.4.1 Expressions

During a conversation, it is an added benefit to understand the emotions of the other individuals. Knowledge of another person's mood can be an incredible asset during an engagement and can in fact lead to a more effective discourse between those involved. This toolbar allows an individual using the CAIRO client to change his/her facial features within the collaboration interface to convey to other clients his/her feelings at a particular instant in time. Currently, the toolbar consists of a fixed set of images that are either cartoons or simple drawings, which could be expanded to allow clients to take various pictures of themselves.

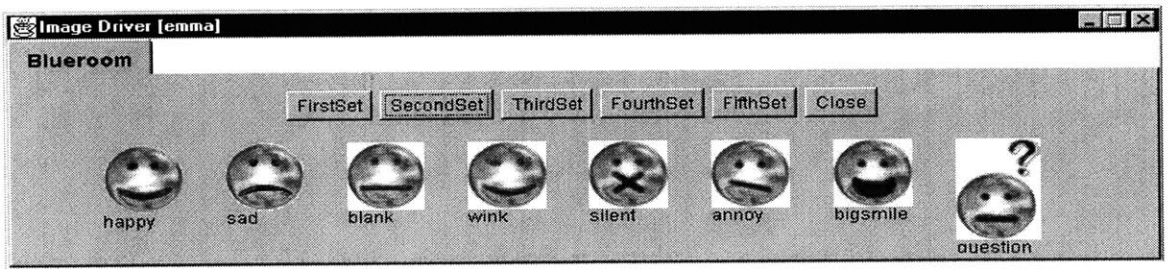

Figure 6-8. Snapshot of Expressions Tool

In addition, the system has been programmed to recognize a series of emoticons in the text driver. Therefore, a participant can simply use one of the emoticons in his/her text driver and CAIRO automatically recognizes it and updates the user's image.

6.1.2.4.2 Casual Contact

At the same time, a passive communication method, termed "casual contact", was added to CAIRO in order to create a sense of place as well as part of the Social Feedback

requirement (item ii of the requirements summary). The concept of casual contact originates from the notion of public and private spaces. When a person works in his/her office or home, this location is considered to be a private area where he/she would not want to be disturbed by incoming events. However, once an individual leaves the confines of a private space to, for example a hallway, he/she has entered a public space. In the public space, there exists the possibility of a chance encounter with other people. These encounters may easily lead to a conversation or unplanned collaboration.

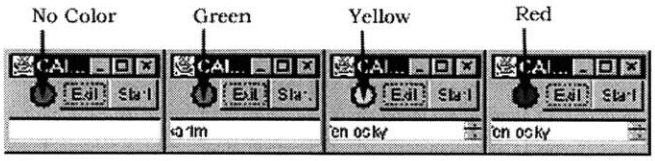

Figure 6-9. Casual Contact

Figure 6-9 displays the four possible states of the casual contact interface. The colored circle reveals the state while the two buttons allow the user to either enter the system or exit it. At the bottom is a list that identifies the users that are on-line. The green circle denotes that the user himself has entered the public domain. A yellow circle signifies that another user has entered the public space. And lastly, a red circle indicates that another person has hailed the user.

6.1.2.4.3 Awareness Driver

The awareness driver was designed and developed as part of the Social Agent for CAIRO [Choudary and Garcia 1999] in order to address the Social Feedback requirement. In a computer based collaborative environment, emotions play a critical role in rational decision-making, in perception and in human interaction [Picard, 1995 & Garcia, Favela, Rodriguez, 1999]. Using affective computing, the participants can create a mental model of their distributed counterparts. From this model, they can get an idea of how their counterparts react in the meeting. This will help simulate the much-needed co-location feeling in the distributed meeting.

This module consists of sensing hardware that detects the physiological changes of the meeting participant and the necessary software modules to interpret the signals. These emotions are then displayed to the other participants through the 3-D Avatars and the Affective Bar or Icon. The primary user interface components of the affective computing module implemented so far are:

6.1.2.4.3.1 Affective Bar

The affective bar gives the affective information (valence, arousal, emotional state) in the form of a continuous plot. The valence is represented by the height (signed value) of the bars and the arousal is represented by the color of the bars.

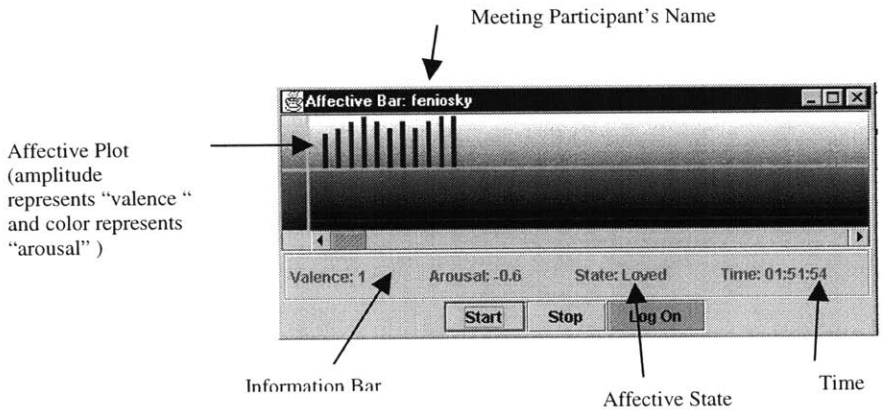

Figure 6-10. Affective Bar

6.1.2.4.3.2 Affective Icons

The affective icons' bar manifests the affective state (valence, arousal, emotional state) using icons containing the meeting participant's images. The color of the band

around the participant's image is representative of the valence and arousal of the user's emotional state – hue represents valence and saturation represents the arousal.

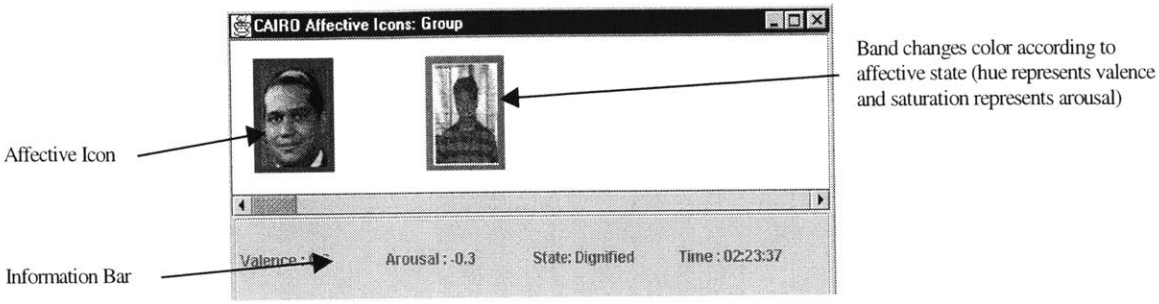

Figure 6-11. Affective Icons

6.1.2.4.4 Three Dimensional Virtual Environment

The three dimensional driver was designed and developed as part of the Social Agent for CAIRO [Solaria and Vedam 1999] in order to create a setting for spatial interaction and visualization of social feedback. With the 3D environment for CAIRO, natural extensions of social interaction and communication are made possible by providing visualization of nonverbal behavior information in order to foster effective communications. Nonverbal behavior is provided through a Nonverbal Behavior Translator (NCT) [Landel, 1999]. A NCT provides methods for capturing, interpreting, translating, and representing nonverbal behavior, namely facial expressions and body movement through camera based data collection, wearable computers and sensors. Once the data is interpreted it is represented through a three dimensional representation of the user or avatar.

This module enables a greater collaborative experience by providing a three-dimensional virtual environment that allows users to visualize movement and social feedback by using gestures for effective communication and social interaction.

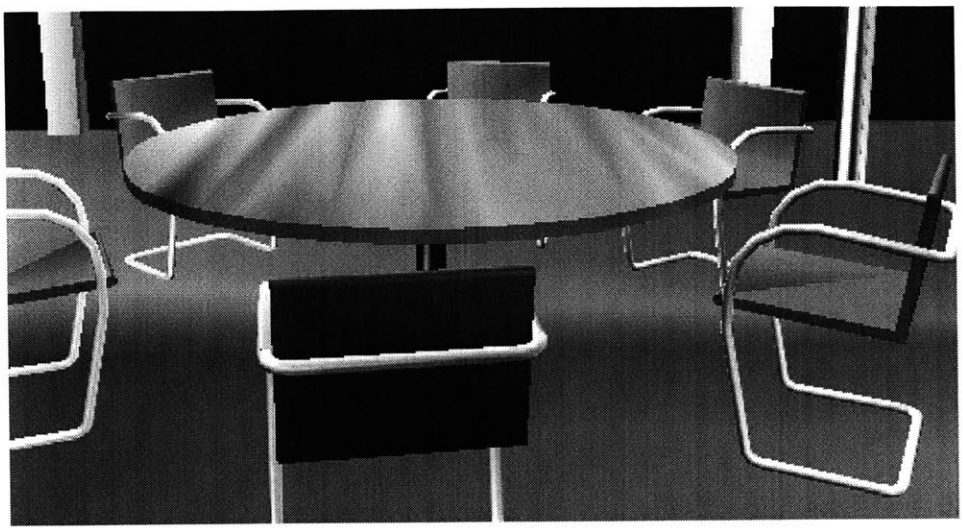

Figure 6-12. 3D Virtual Meeting Room

6.1.3 Conclusions

The main focus of the CAIRO effort has been in the so-called Group Dynamics-Aware Conferencing. The current implementation of CAIRO focuses on the creating a Sense of Place, supporting Spatial Interaction, allowing different Degrees of Engagement and implementing a Floor Control Strategy. Regarding the requirements summary in Chapter 3 the current version of CAIRO has implemented the functionality described in the Process Management, Communication Media and Social Feedback requirements. The system functionality that hasn't been implemented yet includes the ability to support Parallel, Sequential and Reciprocal collaboration behavior, Information Sharing and the architecture is not yet interoperable in heterogeneous networks.

6.3 Virtual Network Computing

The so-called network computer (NC) aims to give users access to centralized resources from simple, inexpensive devices. These devices act as clients to more powerful server machines that are connected to the network and provide applications, data, and storage for the users logged into the system.

The Virtual Network Computing (VNC) project is not the only network computer system. Other thin-client systems include those built around the Citrix ICA protocol (for example, Citrix's Winframe and Insignia Solutions' Ntrigue), those built around Microsofts Remote Display Protocol (for example SCO's Tarantella, and Datalight's ThinSystem as well as Microsoft's own Windows based Terminal Server previously codenamed Hydra) and those built around the X protocol such as Graphon's RapidX and Bridges. The problem with all of these systems is that, unlike X, they use proprietary protocols, so reliable information about them is difficult to obtain. Citrix's ICA protocol is a popular mechanism for remote interaction with PCs, but it appears to be closely tied to the Microsoft Windows GUI, so it may not be an ideal general-purpose remote display protocol. Microsoft has developed its own protocol, T.128 previously known as T.Share, based on the ITU T.120 protocol. The objective of the VNC project is that the protocol, or something similar to it, may become an open cross-platform standard for very thin-client computing.

In the VNC system, server machines supply an entire desktop environment that can be accessed from any Internet connected machine using a simple software NC. Whenever and wherever a VNC desktop is accessed, its state and configuration (right down to the position of the cursor) are exactly the same as when it was last accessed. In addition, VNC allows a single desktop to be accessed from several places simultaneously, thus supporting application sharing in the style of computer supported cooperative work (CSCW).

The technology underlying VNC is a simple remote display protocol. It is the simplicity of this protocol that makes VNC so powerful. Unlike other remote display protocols such as the X Window System and Citrix's ICA, the VNC protocol is totally independent of operating system, windowing system, and applications.

As explained in Chapter 5, the X Window System allows applications to display a user interface on a remote machine. There are, however, several problems with X that restrict its use and, in turn, restrict systems based on it, such as XMX:

A. X requires the display machine to run an X server program. This heavyweight piece of software requires substantial resources, which machines such as NCs and personal digital assistants (PDAs) cannot be expected to run.

B. The X security model makes it inherently dangerous to allow a remote machine to use your display. Accordingly, most system administrators stop X traffic from passing in or out of their sites. Many of the security problems associated with remote displays are discussed in Section 7.2.5.

C. Application startup is extremely slow on high-latency links due to the number of roundtrips performed by a typical application (though there are special proxies that alleviate this problem, such as Low Bandwidth X [Open Group 1997].

In addition to these technical problems, there is also the non-technical problem that X is not Windows, and the world is increasingly Microsoft dominated. Because of the heavyweight components necessary on the client, the X security model, the slow application startup and the lack of Microsoft Windows connectivity make solution based on X less attractive.

6.3.1 The VNC Protocol

The technology underlying the VNC system is a simple protocol for remote access to graphical user interfaces. It works at the frame buffer level and therefore applies to all operating systems, windowing systems, and applications—indeed to any device with some form of communications link. The protocol will operate over any reliable transport such as RS232, firewire, USB, modems and IrDA, anything which gives a reliable 2-way connection. At present TCP/IP is the only protocol implemented, because it's convenient, ubiquitous, and easy to route. The endpoint with which the user interacts (that is, the display and/or input devices) is called the client viewer. The endpoint where changes to the frame buffer originate (that is, the windowing system and applications) is known as the server (Figure 6-13). VNC is truly a "thin-client" system. Its design makes very few requirements of the client, and therefore simplifies the task of creating clients to run on a wide range of hardware.

The display side of the protocol is based on a single graphics primitive: Put a rectangle of pixel data at a given x, y position. At first glance this might seem an inefficient way to draw some user interface components. However, allowing various encoding schemes for the pixel data gives a large degree of flexibility in trading off parameters such as network bandwidth, client drawing speed, and server processing speed. The lowest common denominator is the so-called raw encoding, where the pixel data for a rectangle is simply sent in left-to-right scanline order. All VNC clients and servers must support this encoding. However, the encoding actually used on a given connection can be negotiated according to the capabilities of the server and client and the connection between them. For example, copyrectangle encoding is very simple and efficient, and can be used when the client already has the same pixel data elsewhere in its frame buffer. The encoding on the wire is simply an x, y coordinate. This gives a position in the frame buffer from which the client can copy the rectangle of pixel data. This encoding is typically used when the user moves a window across the screen or scrolls a window's contents. Most clients will support copyrectangle encoding, since it is

generally easy to implement, saves bandwidth, and is likely to be faster than sending raw data again. However, in a case where a client cannot easily read back from its frame buffer, the client could specify that it should not be sent data encoded this way. A typical workstation desktop has large areas of solid color and text. One of our most effective encoding takes advantage of this phenomenon by describing rectangles consisting of one majority (background) color and "subrectangles" of different colors. There are numerous other possible schemes. JPEG encoding could be used for efficient trans mission of still images or an MPEG encoding for moving images. A pixel-data caching scheme could efficiently encode multiple occurrences of the same text character by referring to the first occurrence.

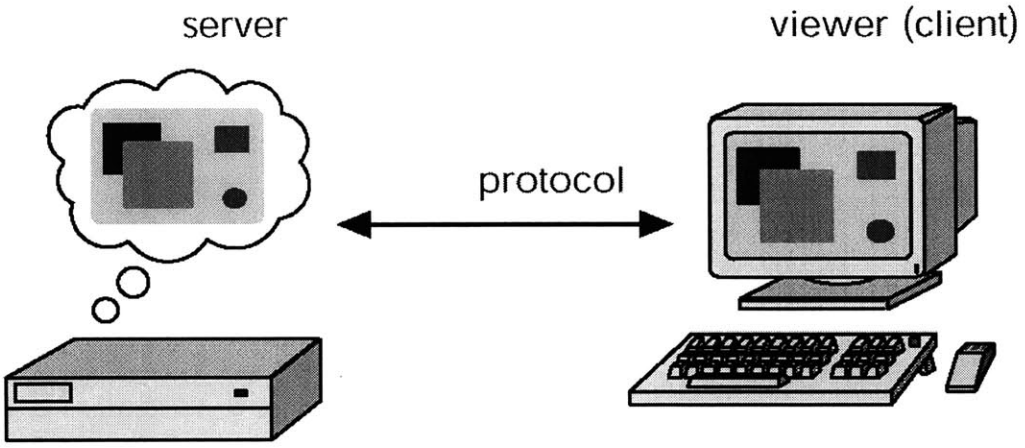

Figure 6-13. Thin Client System [Richardson, Stafford-Fraser, Wood and Hopper 1998]

A set of rectangles of pixel data makes a frame buffer update (or simply, update). An update represents a change from one valid frame buffer state to another. In this sense, an update is similar to a frame of video. It differs, however, in that it usually affects only a small area of the frame buffer. Each rectangle may be encoded using a different scheme. The server can therefore choose the encoding most appropriate for the particular screen content being transmitted and the available network bandwidth. The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. All screen changes since the client's last request are

coalesced into a single update. This gives the protocol an adaptive quality: the slower the client and the network, the lower the rate of updates. On a fast network, for example, as the user drags a window across the screen it will move smoothly, being drawn at all the intermediate positions. On a slower link—for example, over a modem—the client will request updates less frequently, and the window will appear at fewer of these positions. This means that the display will reach its final state as quickly as the network bandwidth will allow, thus maximizing the speed of interaction.

The input side of the VNC protocol is based on a standard workstation model of a keyboard and multi-button pointing device. The client sends input events to the server whenever the user presses a key or pointer button, or moves the pointing device. Input events can also be synthesized from other nonstandard I/O devices.

To establish a client-server connection, the server first requests authentication from the client, using a challenge-response scheme, the client typically requires the user to enter a password at this point. The server and client then exchange messages to negotiate desktop size, pixel format, and encoding schemes. The client requests an update for the entire screen, and the session begins. Because of the stateless nature of the client, either side can close the connection at any time without adverse consequences.

6.3.2 The VNC Viewer

The viewer was designed to be as simple as possible, as indeed it should be for any thin-client system. It requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either writing directly to the frame buffer or going through a windowing system). The available viewers are: an X-based viewer (which runs on Solaris, Linux, and Digital Unix workstations), a Win32 viewer that runs on Windows NT and 95, and a Java applet that runs on any Java-capable browser, a Windows CE viewer that runs on SH3 and MIPS processors and a Palm OS viewer that runs on the IBM Workpad, Palm Pilot Professional, Palm III and Palm V. The images in Figure 2

show a variety of X and Windows desktops being accessed from both Java and native X and Windows viewers.

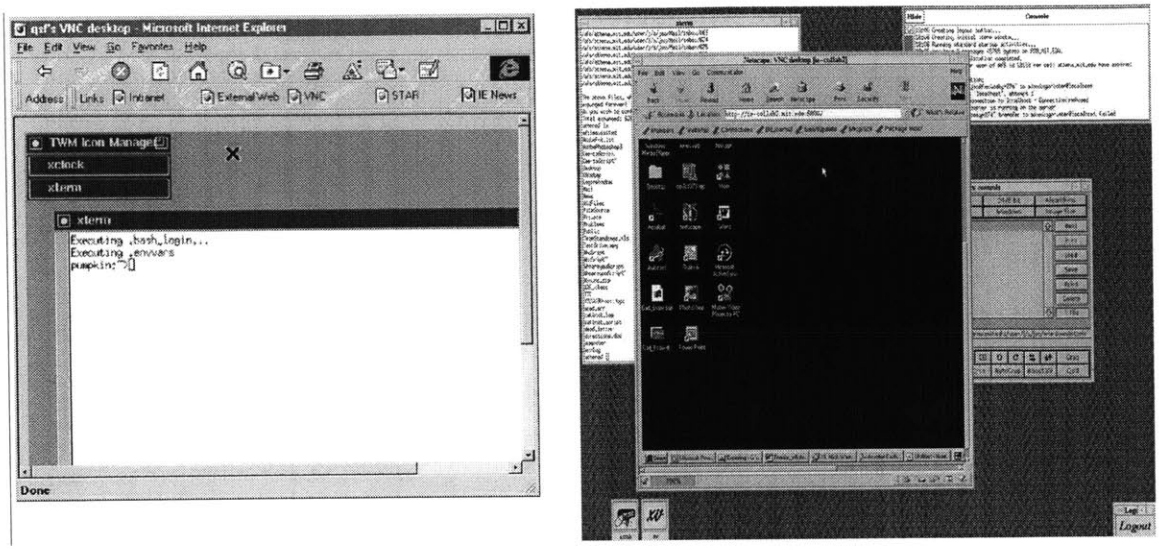

Figure 6-14. Unix Desktop Within Internet Explorer and Windows Desktop Within Netscape on Unix.

6.3.3 The VNC Server

The VNC server architecture is more complex than the viewer. Because the protocol is designed to make the client as simple as possible, it is usually up to the server to perform any necessary translations (for example, the server must provide pixel data in the format the client wants). Servers have been written for two main platforms, X (that is, Unix) and Windows NT/95/2000. The X-based server was the first one developed. A single Unix machine can run a number of VNC servers for different users, each representing a distinct desktop. Each desktop is like a virtual X display, with a root window on which several X applications can appear. The Windows server is more complex since Windows has fewer places to insert hooks into the system to monitor display updates, and the model of multi-user operation is less clearly defined. The current Windows server simply mirrors the real display to a remote client, which means that only a single desktop is available from any one PC running the server [Richardson, Stafford-Fraser, Wood & Hopper].

6.3.4 Conclusion

The protocol's simplicity could allow it to be used on a much wider range of hardware. Consumer electronics devices, such as CD players, usually have a highly specialized user interface and typically employ customized physical display devices. This has traditionally prevented such interfaces from being mobile in the VNC sense of the word. But the usefulness of a remote display protocol can be extended so that users could, for example, bring up the controls for their video recorder on a mobile phone as they drive home from work, use a modem to dial a telephone answering machine and reprogram it through a graphical interface, display their car stereo or GPS receiver as part of the dashboard, regardless of the equipment brand installed. At present, such functions require the displaying device to have detailed knowledge of the remote system and to emulate that system's user interface or some alternative interface that it deems appropriate. For example, you would need a driver for your video recorder, which was designed for your mobile phone's operating system.

A much simpler approach would be to use the interface designed for and provided with the remote device, but to interact with it locally. For this, a set of common "phonemes" is needed with which to construct a variety of GUIs. This is the role that remote display protocols such as VNC can play. It is simple enough to implement cheaply in consumer electronics hardware, yet it can be used to describe the building blocks of most modern user interfaces. With standards such as IEEE-1394 Firewire, USB, and IrDA, the physical interface to connect a variety of devices is provided; with a remote display protocol, a standard for plug-and-play user interfaces can be added.

Imagine walking up to any workstation, connecting your PDA to the USB port, and having the PDA applications instantly available on the workstation screen, or plugging your PDA into your car and having the engine management unit display servicing information on the PDA's screen. And imagine that this works for any workstation, any PDA, any car. The engine management example illustrates an important point: A standardized GUI protocol allows devices that have no physical display of their own to

provide graphical information when such a display becomes available to them. Your PDA could, perhaps, shrink to the size of a pen if it could access a display and keyboard through an IrDA link. And yet this "microPDA" could still display PowerPoint-style presentations when in the vicinity of an LCD projection panel or a large TV .

This model is very similar to the Web, where services without an I/O capability of their own wait for a user to provide one in the form of a Web browser. The success of this strategy has led to embedding HTTP daemons in printers, switches, routers, and other devices. But to be a Web server, a device must at least have a TCP stack and an IP address. And to be a Web browser requires at least the ability to render fonts and parse HTML. In contrast, a remote display protocol requires only a reliable transport medium and the simplest of display capabilities. And while a page of HTML will generally require the transmission of fewer bytes than its remote display protocol equivalent (i.e. the VNC protocol), the latter is infinitely more flexible [Richardson, Stafford-Fraser, Wood & Hopper 1998].

6.3.5 Summary

After discussing the most important features of the CAIRO system from MIT and the VNC applications developed at ORL, an integration of the VNC application into the CAIRO system would greatly enhance the shared workspace capabilities of the current collaboration system. The MIT research has focused on how to facilitate the meeting experience itself with its underlying control structures, while the VNC applications have focused more on the specifications and implementation in order to present remotely the user interface.

Chapter 7

7 Broadband Collaborative ASP

In this chapter, the main considerations of the design of a broadband collaborative application service provider are elucidated. The research process is comprised of a literature review, case studies, scenario illustration, requirements analysis (Chapter 3), exploration of different strategies (Chapter 5), the development of a final solution (Chapter 7) and the subsequent results (Chapter 8) along with their relationship to the final objective are described.

After reviewing the CAIRO system and the VNC project, some considerations for the implementation of a shared workspace must be outlined. The traditional client/server architecture of CAIRO and the accompanying deployment model established by distributed PC-based LAN technology fails to accommodate the needs of multi disciplinary teams working in heterogeneous networks. In order to fulfill the Open Architecture requirement, alternative methods other than traditional deployment of applications are considered to maintain platform independence.

7.1 Design Considerations

Seven practical considerations dominate the design of the broadband collaborative application service provider. These concepts are represented in Figure 7-1 as the axes of a multi-dimensional space. They are:

1. Meeting Protocols: The overhead for conducting meetings might be drastically reduced by having computers handle some of the administrative tasks during meetings such as floor control.

2. Cross-Platform Compatibility: Distributed teams typically employ large, heterogeneous information technology infrastructures, different member organizations use incompatible software components, platforms and devices so it must be possible access a cross software, cross-platform and cross-device shared workspace.

3. Database Integration: Sharing persistent objects implies the storage of these objects in a database in order to obtain secure, reliable communications and group memory.

4. Annotation: To provide awareness of other members' actions and to provide an additional communication channel while sharing an application.

5. Standards Compliance: In order to guarantee users participating in conferencing session interoperability over different types of networks, connections and software vendors.

6. Security: A comprehensive security solution would require secure authentication, cryptography for all communication channels, and fine-grained control over the actions remote users can execute on the shared machines.

7. Concurrency Control: In order to avoid consistency problems when multiple authors work on the same file, the system should offer Event Notification Control, Fine-

grained Notification, User-Controlled Locking, Persistent Collaboration Information and Version Control.

The following subsections provide further explanation of the seven dimensions and how they pertain to the investigation in this research.

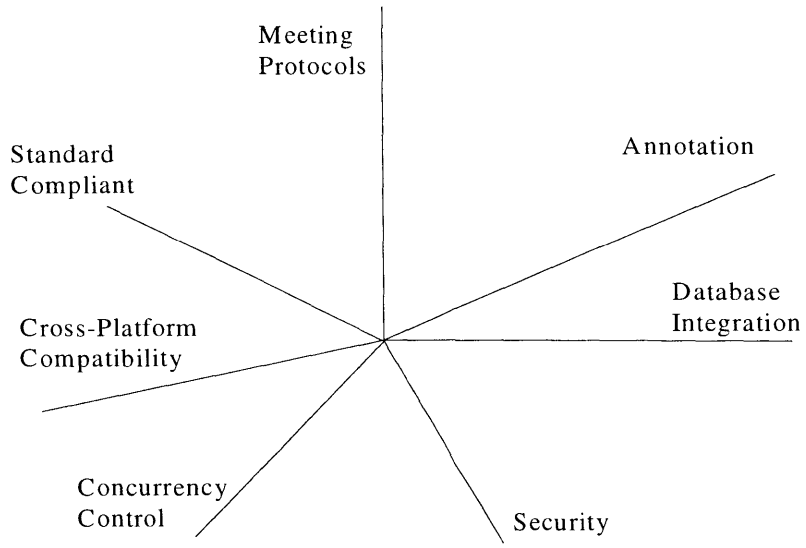

Figure 7-1. Seven Dimensions of Design Consideration

7.1.1 Meeting Protocols

The overhead for conducting meetings might be drastically reduced by having computers handle some of the administrative tasks during meetings. For shared workspaces, the amount of data generated by multi-user input devices can be quite significant. A test run at MIT using seven adjacent CAD users in a Freestyle session, generated enough input events to overload the server and create considerable lag among the viewers, adding to the confusion of who was the participant that was currently drawing since the viewers were not synchronized.

An implementation of the meeting control structure was performed for the collaborative ASP. The results are shown in Figure 7-2. The meeting protocols were

implemented for the so-called lecture style meeting that resembles a classroom type of interaction. In the case shown below a user who doesn't have permission to modify the drawing is not allowed to do so by locking the application area and displaying a warning dialog box when the user clicks on the application with the pointer.

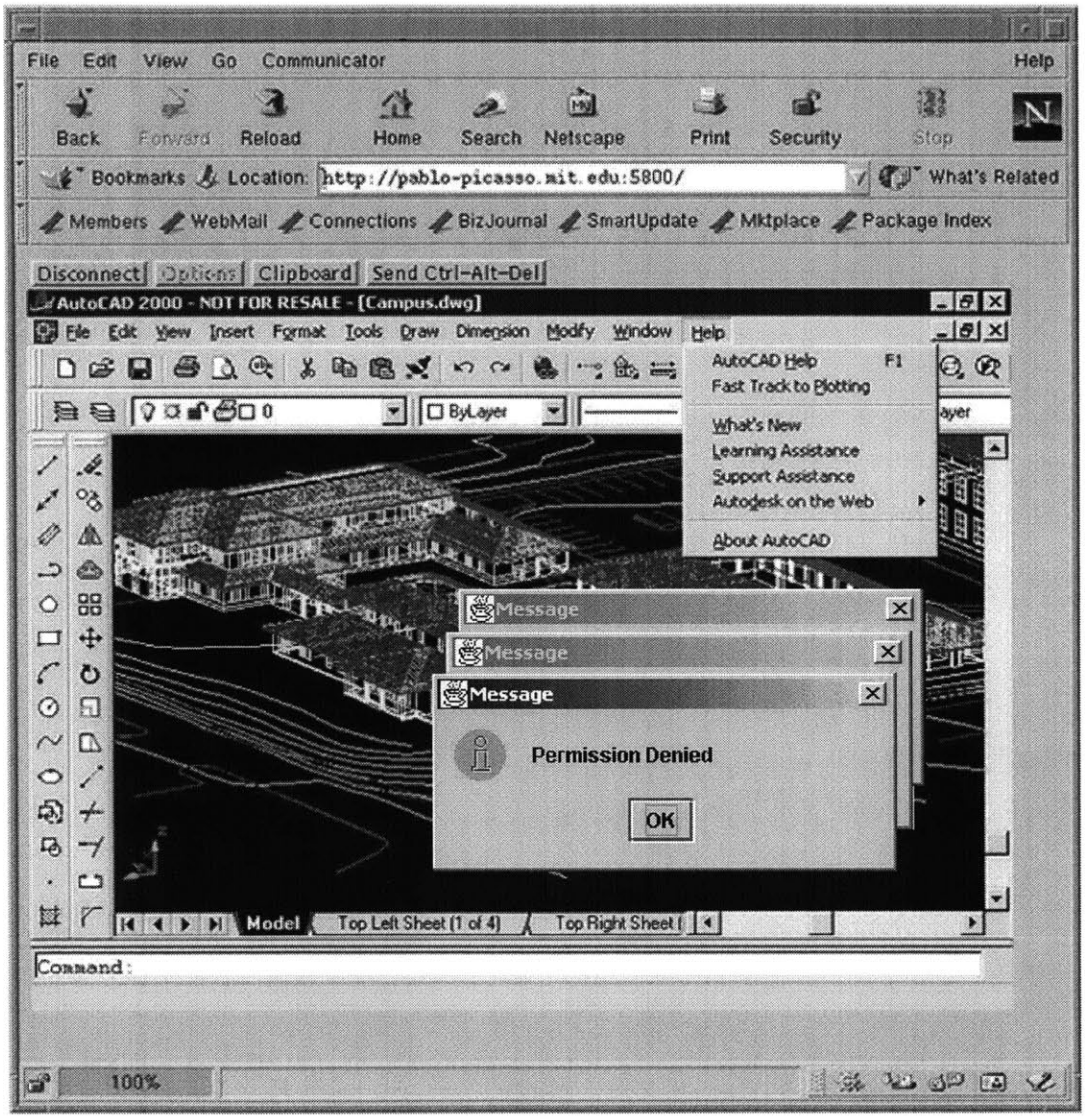

Figure 7-2. Implementation of Meeting Control Structure for Lecture Style

7.1.2 Cross-Platform Compatibility

With every attempt to reduce the number of different kinds of computers and operating systems available in the market, there still exists extensive heterogeneity in hardware and software. As some organizations attempt to standardize on Windows NT, others continue to advocate Unix platforms. At the same time, handheld, palmtop, and wearable computers with their own unique operating systems are beginning to permeate society. Due to the fact that distributed teams typically employ large, heterogeneous information technology infrastructures, different member organizations use incompatible software components and changes in these environments occur at an ever-increasing rate, it must be possible access a cross-platform shared workspace.

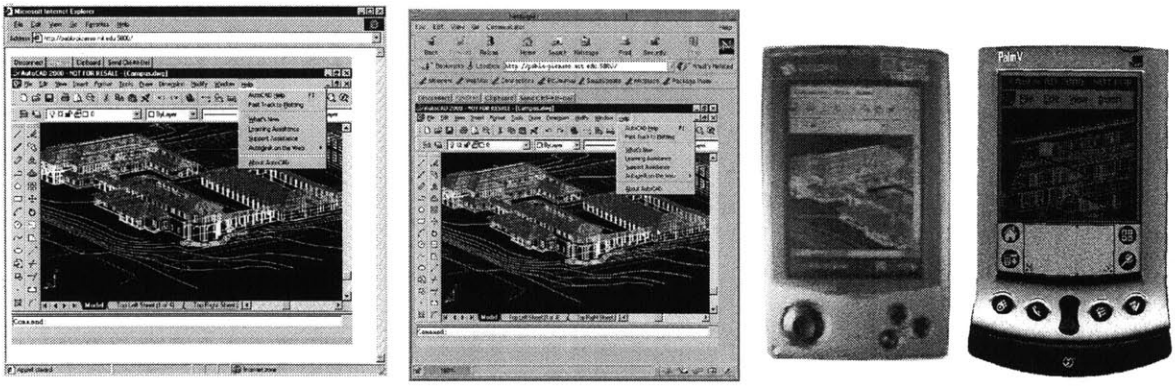

Figure 7-3. Multiple Client Devices: Windows, Unix, Windows CE and Palm

This issue was addressed by using network computer technology such as VNC. Because of the simplicity of the VNC protocol, its design makes very few requirements of the client, and therefore simplifies the task of creating clients to run on a wide range of hardware. The client display side of the protocol is based on a single graphics primitive: Put a rectangle of pixel data at a given x, y position. Even though this may seem inefficient, it allows keeping the capabilities of the client to a minimum. And by providing encoding schemes for the pixel a large degree of flexibility is obtained, especially with respect to client drawing speed. The lowest common denominator for all client devices is the raw encoding, where the pixel data for a rectangle is simply sent in

left-to-right scanline order. This is the only encoding that all VNC clients are required to support.

7.2.2 Database Integration

The database connection fulfills two requirements: secure, reliable communications and group memory. These two purposes are explained below.

During a collaboration session, if a message from one client to another is somehow lost, the two viewers will no longer be synchronized. Over time, this error may propagate to produce contaminated results. The system depends on the overall reliability of its network. With an event-passing paradigm, lost information cannot be retrieved at a central repository because it simply does not exist. To permit resynchronization after reconnection, a database is necessary. Each participant's view is synchronized with the information in the database. When an individual loses his/her connection, the changes that were made during the lapse in communication are reflected as soon as the client is reconnected to the system.

In addition, the database is a site where data can be stored. The existence of an underlying database enables data to persist from one session to another. The users can query the database for information about the session regarding starting time, total time spent, application involved, etc. This repository can help designers understand better the process, conflict resolution, event notification and object locking.

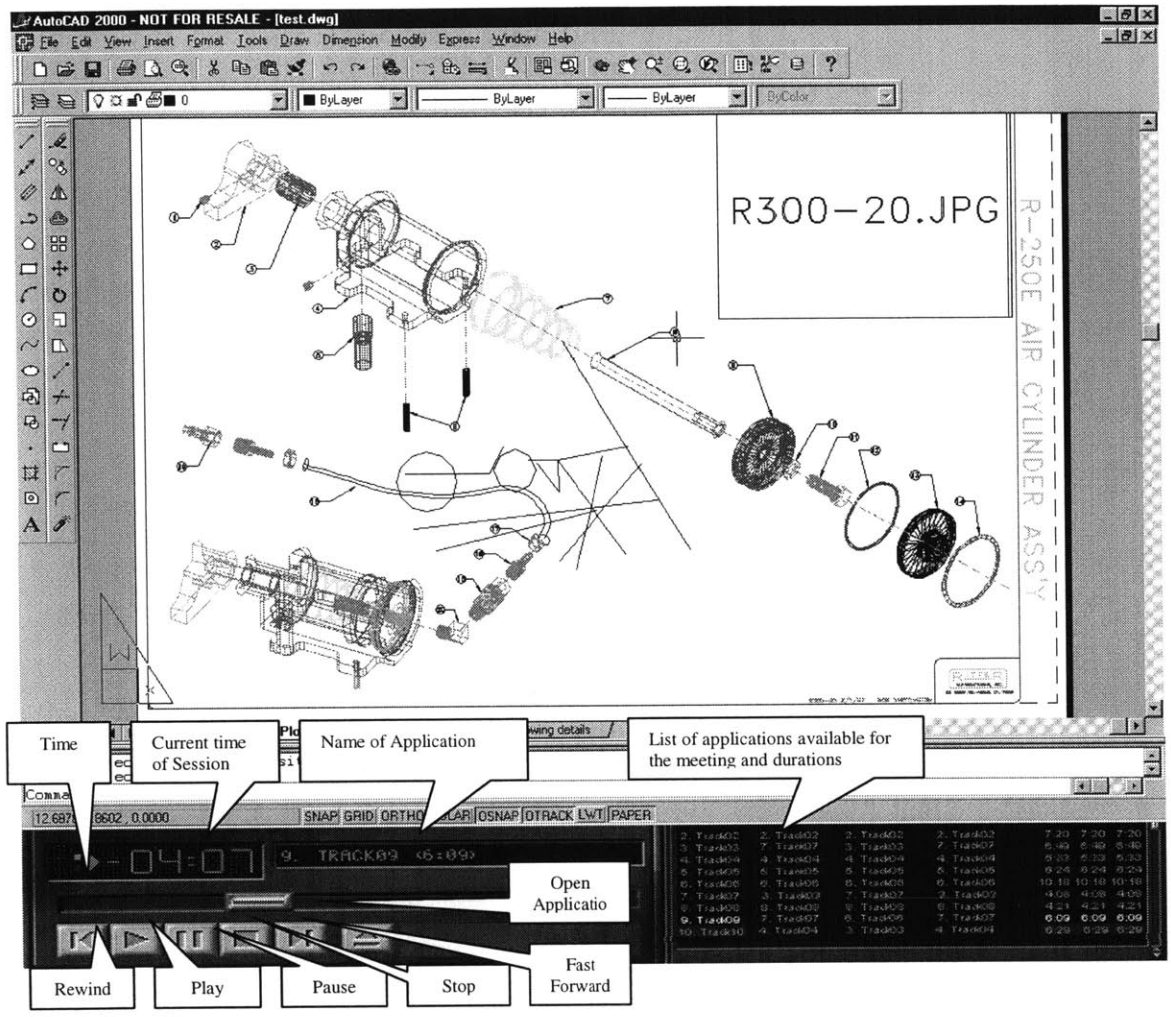

Figure 7-4. Database Query Interface

7.2.3 Annotation

An important aspect outlined in the requirements is the process management and minimizing meeting overhead. One of the key issues is to provide awareness of other members' actions and to provide an additional communication channel while sharing an application. An application launched through the CASP model allows its users to communicate graphically on top of the shared data through the CASP annotation board. The main features of the CASP annotation board are:

Multi-Document Interface. The CASP annotation board contains a multi-document interface such that a user can work on more than one annotation document at a time. A document can be created, opened or saved at any time. Also, figures can be cut/copied/pasted between annotation documents. The user can browse through multiple annotation documents with a tab.

Variety of Figures Available. A selection of figures comes with the whiteboard. A user can easily construct rectangles, lines, arrows, curves, free-hand drawings, images and text within each whiteboard document.

Figure Properties Modification. A user can alter a figure's properties at any time. For instance, a rectangle could be modified such that its color is red and its outline is a thick, blue, dashed line. The appearance of text could also be adjusted by changing its font, font size, and color. Figures can also be grouped/ungrouped along with being ordered to appear in front of or behind other figures.

View Adjustments. The user can adjust each annotation document's view such that it can be scrolled horizontally or vertically and zoomed in or out.

File Opening/Saving. A user can open or save a document anytime. Annotation documents are saved in XML [eXtensible Markup Language] document format. In essence, the whiteboard is an XML browser, with custom defined style sheets [CAIP's XML Whiteboard].

The annotations capabilities including text, underlining, pointers and labels are shown in Figure 7-5.

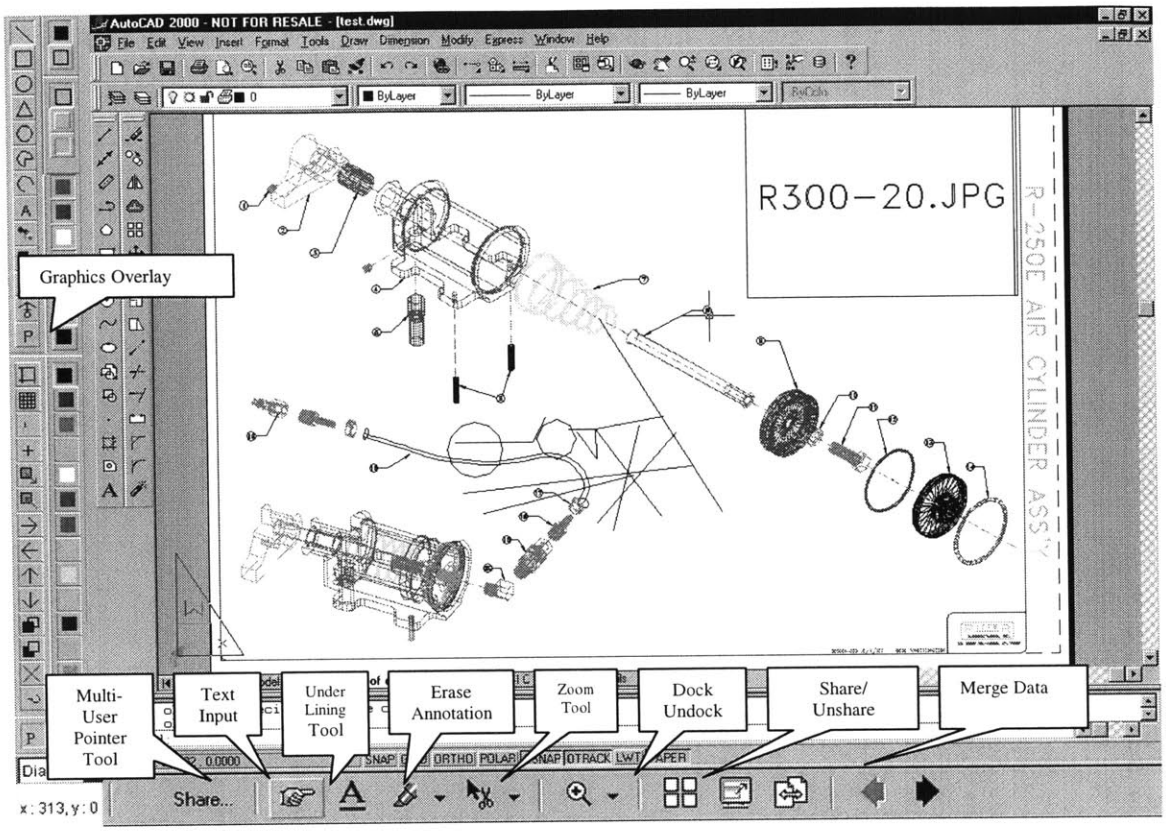

Figure 7-5. Annotation Interface

7.2.4 Standard Compliance

The Broadband Collaborative ASP must base its services on various standards from the International Telecommunications Union [ITU 2000] and the Internet Engineering Task Force [IETF 2000] in order to guarantee users participating in conferencing session interoperability over different types of networks, connections and software vendors. The International Multimedia Teleconferencing Consortium [IMTC 2000] promotes interoperability and is a good resource for ITU specifications. The ITU T.120 standard is made up of a suite of communication and application protocols developed and approved by the international computer and telecommunications industries for use in teleconferencing applications. By complying with the T.120 standards, Broadband Collaborative ASP users can participate in conferencing sessions over different types of networks, connections and software vendors.

The T.120 standard covers document conferencing (data sharing) and provides protocols for establishing and managing data flow, connections and conferences; other standards in the suite cover specific facilities for data conferencing. T.126 specifies multipoint still image and annotation for whiteboard applications. T.127 covers multipoint binary file transfers. The standard that is of most concern to the Broadband Collaborative ASP is T.128. This standard was proposed by Microsoft as an addition to the T.120 standard and was accepted by the International Telecommunication Union, Telecommunication Standardization Sector (ITU-T). T.128 specifies the program sharing protocol, defining how participants in a T.120 conference can share local programs. Specifically, T.128 enables multiple conference participants to view and collaborate on shared programs. The collaborative ASP architecture must be T.128 compliant in the case that users might need to share their desktop or applications that are installed on the local hard drive instead of those available on the application server.

7.2.5 Security

The T.120 data conferencing systems appear difficult to accommodate with firewall packet filters and mayor features of this put the sharing server and all its resources, including its LAN, in serious jeopardy. The standard ITU T.120 services appear to not pose much risk to the sharing or the client systems. The shared whiteboard, which is part of the standard, allows exchange of information but no change of information can be made by a remote user to a shared system or other remote user. The file transfer standard will allow changing the destination host but it is low risk since the received files can be assigned to a specific directory. If senders could target some other directory they could overwrite critical system files. This is the danger posed by the collaborative application-sharing feature.

The feature that poses the most serious risk to network security is the collaborative application sharing. If remote users can take control of the application, the risk here is that any remote user has access to the full power offered by the application. If

the program can write to the hard drive (e.g. Word or Netscape) then the remote user can save files to disk, possibly over writing critical system files. Other scenarios include hostile macros (like existing mail-borne macro viruses), downloading and executing destructive ActiveX or other applications with a web browser, inserting command.com into Word to get a command prompt and downloading network sniffers to harvest passwords. This gives remote users access to unprecedented destructive power on the shared system. From a Word document it is possible to remotely be able to get a command prompt and delete files from the sharing Windows NT system in about 10 seconds, probably not enough time for a network administrator to notice [Shenton 1996].

7.2.5.1 Application Security: Too Much Remote Control

There may be ways of deploying a collaborative ASP that reduces the risk it poses to the user and the enterprise. These rely on network-oriented approaches to hide the users from the attacks rather than trying to solve the inherent vulnerabilities of the deployment model.

7.2.5.2 Virtual Private Networks

The term "Virtual Private Network" (VPN) has gotten a lot of press recently and is marketed as the panacea for all networking ills; this is far from the truth. A VPN securely connects two or more disjoint networks over an intervening network. It might be used to connect different branch offices over the Internet, for example. The presumption is that the intervening network is hostile and by extension that the joining networks are secure.

If the networks in question are not secure, then the VPN only connects one unsecured network to another. If, for example, you use a VPN to connect a well-protected network to one with no protection whatsoever, you have just merged the unsecured net into the secure one, presenting all the vulnerabilities of the latter to the former. VPNs are useful only if you are connecting two secured networks, or at least two networks with the same security posture.

With some commonly available VPN-type tools such as Checkpoint's SecuRemote or MS PPTP, the network security improves but the application vulnerability remains unchanged. The reason is that this solution is an attempt to solve an application-level problem with a network-oriented solution. Note also that these solutions are PC-only, thus limited in its application range [Shenton 1996].

7.2.5.3 Encryption

Third party gateways exist which may help improve reliability, robustness, scalability and security. However, without end-to-end security it is possible for crackers on the remote end to infiltrate a "friendly" network or machine and simply use the gateway as a bridge inside a firewall and attack protected machines as described earlier. The gateway simply requires crackers to first break into machines we allow to connect to our gateway, without offering a complete solution.

A comprehensive security solution would require secure **authentication**, **cryptography** for all communication channels, and **fine-grained control** over the actions remote users can execute on the shared machines. This needs to happen all the way up to the application layer.

For this research VNC used a random challenge-response system to provide the basic **authentication** that allows you to connect to a VNC server. This is reasonably secure; the password is not sent over the network. Once connected, however, traffic between the viewer and the server is unencrypted, and could be snooped by someone with access to the intervening network. Therefore, due to the **cryptography** requirement, it is important to "tunnel" the VNC protocol through some more secure channel such as SSH.

SSH normally just provides you with a 'Secure SHell', a login window to a remote machine. All traffic is encrypted between the two machines using public key encryption techniques. A modified Java viewer created by Mindbright Technology was incorporated

as an SSH applet client [Mindbright Technology 1999]. For the SSH daemon or server, a private Athena workstation was used.

SSH has another advantage. It can compress the data as well. This is particularly useful if the link between the client and the server is a slow one, such as a modem, but even on a faster network it can help make up for the fact that the encryption takes a certain amount of time and so can slow the link down a little.

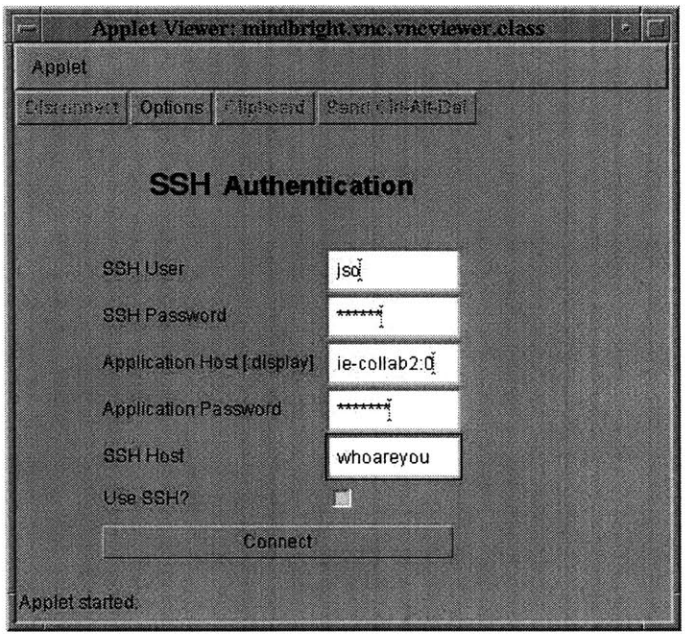

Figure 7-6. SSH Authentication Panel

Fine-grained control over the users actions is the third component necessary to guarantee security. This was not implemented for this research but the most general solution would be to enforce read-only access to the server hosting the applications.

7.2.5.4 Corporate Networks and Firewalls

Unfortunately, the complexity of the implementation as well as compliance with the T.126 standard makes proxying the application's network traffic through a firewall very difficult. Further, the application sharing feature allows remote access to the shared user's

desktop, as explained earlier, allowing arbitrary functions to be run such as file writing, deleting, launching other applications and downloading cracker tools. This application gives excessive control to remote connections making the user's desktop machine and networks resources highly vulnerable, as a result this type of applications are very unpopular among network administrators and security consultants.

7.2.6 Concurrency Control

In order to avoid consistency problems (and confusion) when multiple authors work in the Broadband Collaborative ASP, the system should offer several levels of concurrency control:

Event notification. Users need to be notified of changes made to objects by other users.

Fine-grained notification. The event handler must be able to distinguish between write operations on the application (content) and on other attributes such as annotation.

User-controlled locking. In addition to implicit locking within the scope and control of transactions, users must be able to lock objects during a long update that outlasts several short transactions.

Shared locking. Locked objects must always remain readable, so users can get a common view of the information space.

Fine-grained locking. While one user is updating the application (content) of a node, other users should still be able to annotate on it.

Persistent collaboration information. Locks, events, and other information about the collaboration between users must be stored in a database as discussed in section 7.2.2.

Clients need to be able to recover from server crashes and therefore need to have persistent collaboration information [Kock and Legett, 1993].

7.2.6.1 Locking

User-controlled locking should be *shared, fine-grained* and *persistent.* While a user is writing a (long) node, the ongoing work will be saved frequently so as to not lose a lot of information in the event of a system crash. Meanwhile the user may wish to keep the node locked for the entire duration of the editing session.

7.2.6.1.1 Shared Locking

Allowing locked objects to be read enables users to browse an application containing locked objects. The only conflicts we wish to avoid are write-write conflicts. Being able to read the ongoing work of other writers can be helpful, for instance to view how one writer has created an object another writer wishes to use.

7.2.6.1.2 Fine-Grained Locking

Databases typically provide locking at the object level. Locking at the attribute level reduces the risk of conflicts. While one writer is adding to a node another writer may write to another node.

Since fine-grained locks may be held for only a short time (there's only so much updating one can do on a small part of an object) one may consider queuing lock requests instead of denying them.

7.2.6.1.3 Persistent Locking

Locks should be stored on both the client and the server side. When a server goes down and recovers, the clients should be able to reclaim the locks they had before the

crash. When a client is restarted after a crash it should be able to retrieve the locks from the server.

7.2.6.2 Notification Control

Notification control allows users to be notified of important actions on the shared Collaborative ASP performed by other users. When you view a node currently locked by another user, your view should be updated each time that user saves updates to that node.

Event notification is usually asynchronous. Synchronous notification requires polling, which is very inefficient. When a node changes all readers receive an event, and will update their display asynchronously.

Individual users may wish to subscribe to events (or not). This places selection overhead on the server. For every operation the server must check which client applications have subscribed to that event. This overhead can be placed with the client by making the client ignore unwanted events (and letting the server send them anyway).

7.2.6.2.1 Fine Grained Notification

Notification of all operations on objects and attributes in objects should be supported, including notification of lock, unlock, delete and write operations on a find-grained level. Notification can only be reliably implemented at the same granularity level as locking.

7.2.6.2.2 Persistent Notification

Event information should be kept persistently in the shared workspace, for the same reason as persistent locking. If either the server or the client crashes, both should be able to recover the event subscriptions.

7.2.6.3 Transactions

Transactions in database systems serve three purposes:

1. They are logical units that group operations comprising a complete task;
2. They are atomicity units whose execution preserves the consistency of the database;
3. They are recovery units that ensure that either all the steps enclosed within them are executed or none.

7.2.6.3.1 Short Transactions

Short transactions are needed for saving editing work while continuing a long editing session. Data is locked in different granularities to prevent other users from performing conflicting operations on the same set of data. In databases short transactions use locks to avoid read-write and write-write conflicts. In a shared workspace only write-write conflicts must be avoided.

7.2.6.3.2 Long Transactions

User-controlled locking combined with short transactions is a better solution to long updating sessions than long transactions.

1. The logical, atomicity and recovery units are much smaller in user-controlled locking.

2. User-controlled (shared) locking does not completely prevent other users from getting to resources over long periods of time.

3. In user-controlled locking, users or applications must explicitly lock needed resources. Users must be aware of the multi-user situation.

7.2.6.4 Version Control

Version control provides the following extensions to concurrency control:

1. The existence of multiple versions of objects eliminates the need for write-write synchronization since each write operation occurs in the context of one version and thus cannot conflict with other operations.
2. A version control mechanism preserves the object identity of different versions of collaborative artifacts.
3. A version control mechanism allows several long updating sessions on the same object to run in parallel with the restriction that different branches (in the version history tree) of the object are used. These branches must be merged at some point in time to reflect the updates performed by all users.
4. The integrity of individual contributions to an object is maintained since the complete history of the object is recorded by the version control mechanism.

The biggest problem in using version control to allow write-write conflicts is that not all simultaneous updates to nodes can be merged later on in a deterministic way. In this sense the meeting control structures might provide queues as to how to resolve these conflicts (i.e. precedence of chairman over other participants).

7.3 Summary

This chapter has the discussed the issues involved in building a Collaborative ASP from the standalone applications discussed in Chapter 6. These issues are:

1. Meeting Protocols.
2. Cross-Platform Compatibility.
3. Database Integration.
4. Annotation.
5. Standards Compliance.

6. Security.
7. Concurrency Control.

These seven items have been the main design considerations of the Broadband Collaborative Application Service Provider. Some have been implemented fully such as Security and Cross-Platform Compatibility and most design items have been partially implemented as a proof of concept. This is the case of the Database Integration, Meeting Protocols and Annotation. Integration of the remaining elements into the final design have not yet been studied, these items that remain to be integrated are Standards Compliance and Concurrency Control. The following chapter will outline the main results of this design.

After completing the modifications described in the previous paragraph, an initial prototype system of a shared workspace for that allows distributed collaboration between participants for architectural design purposes was implemented. For a number of technical reasons the VNC system was not integrated into the CAIRO system, but many of the principles and concepts that are present in the CAIRO project were introduced and adapted to VNC.

This prototype served as a first iteration of a Collaborative ASP for shared workspaces. It provides a solid framework upon which new modifications and versions may be built. Furthermore, this prototype can then be used to study the types of interactions that may occur with such a workspace and provide insight on how to further develop the Collaborative ASP.

Chapter 8

8 Results

Upon completion of the prototype, an evaluation of its functionality was conducted. This evaluation should lead to a new iteration of the requirements upon which a new architecture should be designed.

This evaluation will focus on the performance of the clients and their usefulness in a collaborative scenario. In order to evaluate the performance three different clients were explored, the java applet on a desktop computer (Unix and Windows), the Palm native client on a Palm Pilot and the Windows CE native client on a Cassiopeia E-115. Regarding the usefulness of the Collaborative ASP in a virtual team setting a simple pilot study was performed.

8.1 Pilot Study

The pilot study consisted of seven meeting participants engaged in a Freestyle meeting with the same CAD drawing open on each desktop. There was no annotation facilities, database recording or concurrency control available for this session. The application sharing was strictly WYSIWIS.

The first problem encountered was control over the tele-pointer. The tele-pointer is the pointer on the server host to which all viewers have access. If only one viewer moved the local pointer that person would have control over the tele-pointer, but if two or more participants started moving their local pointers, the tele-pointer would respond sometimes to a combination of all three and sometimes to none at all.

The next problem that was noted for this interaction style was that when more than one person introduced a modification in the drawing or even moved the pointer there was no feedback on who was the author of this action. In this sense there should be an awareness mechanism that should inform the meeting participants of what the other users actions are.

Another problem when several participants made modifications to the drawing was that this generated enough input events to overload the server. The server was able to process all the requests made in the order in which they were received but the time needed to finish processing all the events caused the clients to lose synchronization with the server. This problem made the interaction more confusing regarding which participant was currently in control of the tele-pointer because the participants had no feedback of this information and adding to the problem, the viewers were not synchronized. For example, a participant might be thinking that what is currently being drawn in the screen are his/her when they are actually another participants actions done a few seconds before.

This case study is applicable for a meeting interaction based on the Freestyle control structure. Among the different control structures discussed in the CAIRO

description, this is the control structure that is could cause the most confusion in terms of joint ownership and manipulation of "work entities" or objects of common concern.

8.2 Performance

The performances of the different desktop clients were very similar among each other and the performance of the handheld devices were also similar among each other. Between the desktops and the handheld devices there were noticeable differences. So for the purposes of this review all desktop computers regardless of the platform will be simply described as desktops. And the two handheld devices used will be referred as personal digital assistants.

8.2.1 Desktop Clients

In general both platforms have provided satisfactory performance with a minimum amount of lag. Comparing the performance of the desktop platforms has shown that it is faster to use a Windows box to view a Unix machine rather than the other way around. This is chiefly because Windows generally works better as a client than as a server, and also because PC graphics cards are often better than those in Unix workstations. The performance of the handheld devices was not as satisfactory for reasons described in the following sections.

8.2.1 Personal Digital Assistants

For this research two PDAs were used, a Windows CE device running on a Cassiopeia E-115 with 131 MHz and 32 MB and a Palm Pilot V with 2 MB of RAM. The connections used for the PDAs were PPP connections to MIT's Tether service using vendor specific modems. From some preliminary tests, the Windows CE device had better performance in processing time and display resolution of CAD drawings. During the testing of these devices several limitations became apparent for both devices. None of

these limitations were due to the software but mostly because of limitations that are inherent to the type of hardware these devices have. Further descriptions of these limitations are provided in the following sections.

In their paper, "Multimedia Client Implementation on Personal Digital Assistants", Markus Lauff and Hans-Werner Gellersen [1997] discuss the limitations of using handheld computing devices as network clients. They attribute the difficulty in using network applications effectively to three main reasons:

1. Display Limitations

2. Memory and Computational Power Limitations

3. Network Connectivity

8.2.1.1 Resource Limitations

Under the heading of resource limitations, we encounter problems such as small display, limited color depth, small memory, and weak computing power as issues that must be dealt with when trying to develop network applications on PDAs. The following subsections address these issues more elaborately.

8.2.1.1.1 Display Limitations

The common problem for PDA displays is that the interface display is too large for the screen. In their text, they introduce a number of possible solutions, the most important being:

- The first possible resolution is to use two scrollbars. Although double scrollbars seems like an intuitive answer at first, the disadvantage is that it can be very inconvenient to have to scroll across the display. Thus, since the user cannot see the whole display at the same time, it may be difficult to track one's position

within the interface. This solution requires that the PDA have enough computing power to store the whole display and scroll through the interface.

- Another approach is to reformat the size of the display so that it fits the width of the screen. Unfortunately, with a small screen and a limited resolution the user interface may become illegible.

- Lastly, the display graphical user interface can be replaced with text input if it is not important to the ease of use of the application. This approach requires some sort of heuristic to be developed to determine the relative worth of a user input dialog window.

Each of these solutions elucidates the problems that may be encountered with the small displays available to Personal Digital Assistants. As more research is done in this arena, more problems are sure to arise. These issues should be analyzed further to allow a shared workspace system for collaborative design to account for these problems.

8.2.1.1.2 Memory and Computational Power Limitations

The small memory and low computing power of PDAs also presents a problem. To alleviate the situation, the authors suggest the use of a proxy server to preprocess some of the data. The PDA could send the server information about its dimensions and capabilities. The proxy then redefines the data to fit the capacity of the unit. This approach removes the restriction on the information format as well as the computing power of the PDA. On the other hand, the PDA must be capable of accessing a proxy server.

8.2.1.1.3 Network Connectivity

Another issue with PDAs stems from their ability to connect to networks. The more recent versions of the Palm Pilot (III, V and V) as well as the Windows CE Pocket PCs do have the ability to establish a direct TCP/IP connection to the Internet with the

addition of a peripheral wireless IP link. For the purposes of this research, a Palm III connected directly the Internet through a wireless modem provided by Novatel Wireless was used [Novatel Wireless, Inc. 1999]. The Palm V was connected to a 3Com [3Com Inc, 1998] and the Windows CE device also used proprietary modem [Casio Inc. 2000] connected to a phone line. The best results in terms of connection speed were obtained with the Windows CE device sharing a word processing document or numeric spreadsheet. When the handheld devices were used to share graphic intensive applications such as Cad packages the time spent accessing the data increased dramatically.

8.3 Summary

In summary the performance of the Collaborative ASP in a Freestyle meeting with seven adjacent CAD viewers had several deficiencies that could have been improved if a minimum of the items discussed in Chapter 7 were in place. The most important of them are the meeting protocols, annotation, concurrency control and awareness of other users actions. There weren't any quantitative metrics employed in this pilot study, the only criteria used was common sense while observing what worked and what created confusion for the people engaged in the collaboration.

For the performance of the client devices in terms of traditional computational speed, more thorough observations were possible. Even though there were no quantitative measurements done in this case either, there is no doubt about the order in which the devices ranked.

Chapter 9

9 Conclusion

This research consisted of two main goals. The initial goal was to develop a set of requirements that would need to be fulfilled by a prototype collaboration system and within these requirements to identify a special subset, which were those requirements that applied to creating a shared workspace. The general requirements can be found in Chapter 3 and the specific subset of requirements has been translated into design considerations for the Collaborative ASP, discussed in Chapter 7. These requirements are not exhaustive in any way but could provide a framework upon which various scenarios could have been played out. Regarding the design considerations outlined in Chapter 7, these issues were identified as the most important from personal experience and literature review but also keeping in mind how to leverage existing technology to obtain the original goal. For example, database integration gained precedence over concurrency control due to implementation difficulties of the latter.

The second goal of this thesis was to integrate an existing collaborative application such as CAIRO with a new, shared workspace. This was done successfully

for the case of the multi-user extension described in Chapter 5. But as it stands, the integration with a Shared X solution, to which VNC is related, would not have been fully successful. However, the problems that were encountered during the development of the multi-user extension did shed some light on some of the limitations of CAIRO. With today's object oriented software development techniques, the key to success is to have a software package that is flexible and can be easily integrated with other systems. As was discovered, CAIRO at this stage does not provide this kind of service. Furthermore, the connectivity to third party applications was poor. For this reason, an approach based on integrating the CAIRO's concepts into the VNC system was used instead of the actual systems integration. The most important features CAIRO has to offer at a conceptual level for a successful implementation of a shared workspace are the control structures associated with the different meeting styles.

The next step is to take the architecture presented within this research effort and bring it to full realization. As explained in the summary of Chapter 7 only some of the concepts developed during this research could be implemented into the prototype for the Broadband Collaborative ASP. Many items listed in the requirements are still under development and others have not yet even started to be implemented. Until this first iteration has concluded it will not be possible to provide an environment in which shared workspaces for CSCW can be enabled and studied.

Finally, in the not too distant future, more research must be conducted into the concurrency control. Although the concurrency control is briefly touched upon in this document, the details need to be identified more concretely.

Bibliography

[Anumba, C. and Duke, A. 1999] "Telepresence in Concurrent Lifecycle Design and Construction", to be published.

[Anumba, C. and Duke, A. 1997] "Telepresence in Virtual Project Team Communications", Proc. ECCE Symposium on Computers in the Practice of Building and Civil Engineering, Lahti, Finland, 3-5 September, pp 80-84.

[ASP Industry Consortium 2000] An international advocacy group formed to promote the application service provider industry by sponsoring research and articulating the strategic and measurable benefits of this evolving delivery model. http://www.aspindustry.org, December 1999.

[Bazik, J. 1999] Author of XMX an X protocol multiplexor, Manager of Research and Educational Software, Department of Computer Science, Brown University.

[Benjamin, K. 1998] "Defining Negotiation Process Methodologies for Distributed Meeting Environments." MEng Thesis, May 1998, Massachusetts Institute of Technology.

[Brooke, J. 1995] "User Interfaces for CSCW Systems." CSCW in Practice: An Introduction and Case Studies. Pp.23-30. Edited by M. Tan and R. The, Singapore: Centre for Advanced Studies in Architecture, University of Singapore, 1995.

[CAIP's XML Supported Whiteboard] In the DISCIPLE Project a WhiteBoard Java Bean has been developed using the Multi-Document Editor Framework. This Bean provides a multi-document interface that allows a user to construct complex 2D graphical drawings into one or more documents (spaces). http://www.caip.rutgers.edu/disciple/whiteboard/

[Cicognani, A. and Maher, M. 1997] "Models of Collaboration for Designers in a Computer-Supported Environment." In Third International IFIP WG5.2, Workshop on Formal Aspects of Collaborative CAD, Conference Proceedings, Jenolan Caves, Sydney, edited by M.L. Maher, J.S. Gero, and F. Sudweeks, IFIP, KCDC, pp.99-108, 16-19 February 1997.

[Citrix Systems Inc. 1999] Developer of application server software as well as consulting services for enterprises. Citrix application server software and services enable organizations to run applications on servers that can be accessed from a variety of

client devices. Server-based Computing White Paper, Citrix Systems, Inc. (1999) ftp://ftp.citrix.com/doclib/SBCWP.PDF (http://www.citrix.com, November 1999)

[Cochrane, P., Heatley, D. and Cameron, K. 1993] "Telepresence - Visual Telecommunications into the Next Century", Proc. 4th IEE Conference on Telecommunications, Manchester, pp 175-180, April 1993.

[Condon, C. 1993] "The Computer won: Cooperation, Conflict and the Ownership of Information." In CSCW: Cooperation or Conflict? pp.171-185. Edited by Steve Easterbrook, London: Soringer-Verlag, 1993.

[Choudary, K. and Garcia, O. 1999] Software Design Document, Affective Computing Group, Distributed Software Engineering Laboratory (DISEL), MIT, March 1999.

[Extensible Markup Language] The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web. http://www.w3.org/XML

[Fluckiger, F. 1995] Understanding networked multimedia: applications and technology. London, New York. Prentice Hall, 1995.

[García, O., Favela, J. and Rodríguez, A. 1998] Emotional Awareness in Collaborative Systems. String Processing and Information Retrieval Symposium & International Workshop on Groupware. Cancun, Mexico. 21 - 24 September, 1999.

[Hussein, K. 1998] "Computer Supported Interaction in Distributed in Distributed Design Teams" PhD Thesis, Massachusetts Institute of Technology, May 1998.

[IETF 2000] The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. http://www.ietf.org

[IMTC 2000] The International Multimedia Teleconferencing Consortium, Inc. (IMTC) is a non-profit corporation comprising more than 150 companies. The IMTC's mission is to promote, encourage, and facilitate the development and implementation of interoperable multimedia teleconferencing solutions based on open international standards. http://www.imtc.org

[ITU 2000] The International Telecommunication Union, headquartered in Geneva, Switzerland is an international organization within which governments and the private sector coordinate global telecom networks and services. The ITU is a leading publisher of telecommunication technology, regulatory and standards information. http://www.itu.int

[JCraft 2000] JCraft is a service/product company that leverage Internet technology to manage electronic information. Especially, JCraft dedicates to the Java platform,

one of Jcraft's products is WiredX. Jcraft, Sendai 980-0804 Japan. http://www.jcraft.com

[Johannsen, A., Haake, J. and Streitz, N. 1996] Tellecollaboration in Virtual Organisations – The Role of Ubiquitous Meeting Systems. Arbeitspapiere der GMD 974. February 1996.

[Jones, O. 1993] Multidisplay Software in X: A Survey of Architectures, The X Resource, Issue 6, O'Reilly & Associates, 1993

[Kareem, A. 1998] Defining Negotiation Process Methodologies for Distributed Meeting Environments. Master of Engineering in Electrical Engineering and Computer Science Thesis, MIT, May 1998.

[Landel, G. 1999] Cross Cultural Computer-Supported Collaboration, Master of Engineering in Civil and Environmental Engineering Thesis, MIT, June 1999.

[Lauff, M. and Gellersen, H. 1997] "Multimedia Client Implementation on Personal Digital Assistants." Interactive Distributed Multimedia Systems and Telecommunication Services, September 1997, pp. 283-295. Darmstadt, Germany.

[McGrath, A. 1998] "The Forum" Siggroup Bulletin, Vol 19, No 3, December, ACM Press, New York. Pages 21-25.

[McKie, S. 1999] "Outsourcing with ASPs in the Internet Age", Business Finance: Bottomline Solutions for Financial Executives, Duke Communications International, November, 1999 Loveland, Colorado.

[Mindbright Technology AB 2000] Mindbright specializes in products and services within the areas of datacommunication, security and distributed systems. http://www.mindbright.se

[Moore, C. 1986] The Mediation Process : Practical Strategies for Resolving Conflict. Jossey Bass, San Francisco, CA, 1986.

[Novatel Wireless, Inc. 1999] "Corporate Profile." URL: http: //www.novatelwireless.com/html/profile.htm. Novatel Wireless, Inc., May 13, 1999.

[Open Group 1997] "X11R6.3 (Broadway) Overview," http://www.opennc.org/desktop/x-window-system/broadway.htm#lbx 1997.

[Peña-Mora, F., Anumba, C., Solari, J. and Duke, A. 2000] An Integrated Telepresence Environment for Collaboration in Construction, Special Issue on Computer Aided

Engineering, Engineering with Computers, An International Journal for Simulation-Based Engineering, 2000.

[Peña-Mora, F. and Hussein, K. 1998] Interaction Dynamics in Collaborative Civil Engineering Design Discourse: Applications in Computer Mediated Communication. Microcomputer in Civil Engineering Journal, 1998.

[Peña-Mora, F., Ali-Ahmad, W., Chen, H., Hussein, K., Kennedy, J., Soibelman, L., and Vadhavkar, S. 1996] "The Da Vinci Agent Society Initiative Progress Report as Fall 1996." *IESL Report No. 96-06*, Intelligent Engineering Systems Laboratory, Engineering System Group, Henry L. Pierce Laboratory, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, November, 1996.

[Peña-Mora, F. and Hussein, K. 1999] "Proactive Meeting Management for Distributed Collaborative Design". Advances in Engineering Software, 1999.

[Picard, R. 1995] Affective Computing, MIT Media Laboratory Perceptual Section Technical Report No 321,1995.

[Picard, R. and Healey, J. 1997] Affective Wearables, MIT Media Laboratory Perceptual Section Technical Report No 467,1997.

[Richardson, T., Stafford-Fraser, Q., Wood, K., and Hopper, A. 1998] "Virtual Network Computing", IEEE Internet Computing, Vol.2 No.1, Jan/Feb 1998 pp33-38.

[Saad, M. and Maher, M. 1995] "Exploring the possibilities for Computer Support for Collaborative Designing." In The Global Design Studio. pp.727-738. Edited by M. Tan and R. The, Singapore: Centre for Advanced studies in Architecture, University of Singapore, 1995.

[Schefström, D. 1999] Internet and Distributed Multimedia Course, Center for Distance-Spanning Technology, Luleå University of Technology, Sweden, 1999.

[Sen, S. 1999] Computer Supported Interaction in Distributed Design Reports, Research Progress Report, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, October, 1999.

[Sharples, M. 1993] "Adding a Little Structure to Collaborative Writing." In CSCW: Cooperation or Conflict? pp.51-67. Edited by Steve Easterbrook, London: Soringer-Verlag, 1993.

[Shenton, C. 1998] "NetMeeting Security Concerns and Deployment Issues", National Aeronautics and Space Administration, white paper, 1998.

[Solari, J. and Vedam, P. 1999] Software Design Document, Three Dimensional Virtual Environment Group, Distributed Software Engineering Laboratory (DISEL), MIT, March 1999.

[Su, C. 1998] Distributed Software Design for Collaborative Learning System Over the Internet, Master of Engineering in Electrical Engineering and Computer Science Thesis, MIT, May 1998.

[Sun Microsystems, Inc. 1999] "Java Development Kit." URL: http://www.javasoft.com/products/jdk/1.2/index.html Sun Microsystems, Inc., May 13, 1999.

[Walker, G. and Sheppard, P. 1997] "Telepresence – The Future of Telephony", BT Technology Journal, Vol. 15, No. 4, October, pp 11-18.

[Wiil, K. and Leggett, J. 1993] Concurrency Control in Collaborative Hypertext Systems, 5th ACM Conference on Hypertext, Seattle, November 1993, pp 14-24.