CSCW - WCSC: Computer-Supported Cooperative Work - What Changes for the Science of Computing

Marilyn M. Mantei

Dynamic Graphics Project
Computer Systems Research Institute and
Department of Computer Science
University of Toronto
6 Kings College Road
Toronto, Ontario, Canada M5S 1A4
416-978-5512 FAX: 416-978-4765 E-mail: mantei@dgp.utoronto.ca

Abstract

Computer-supported cooperative work environments change some of the underlying beliefs about solutions that have been built for distributed computing. Electronic mail and file transfers have worked efficiently and effectively by breaking the information to be transferred into packets and reassembling the packets at the destination. This is not a viable solution for handling shared real time drawing or writing. Integrity has been maintained by locking out portions of a database until an update is completed. Such lockout is not always suitable in a groupware interface. Other solutions are necessary for resolving conflicts. Client-server architectures have worked well for managing distributed processing but replicated architectures, coupled with their fragility and synchronization problems are needed for groupware products in order to preserve acceptable local response time. The addition of multi-media to the environment complicates the problem more. Control of analog video requires a client-server environment which can build in intolerable delays as distances between communicating parties increase. This paper approaches design criteria for shared software from the human side and points out profound architectural problems that need to be solved if multi-media cooperative work environments are to function effectively.

Resumé

Les environments de tâches coopératives médiatisés par ordinateur changent certaines des croyances sur les solutions qui ont été mise en oeuvre pour l'informatique distribuée. Le courrier électronique et le transfert de fichiers ont fonctionnés avec succès et efficacité en séparant l'information en paquets qui sont réassemblés a` l'arrivée. Cette solution n'est pas viable pour traiter la composition coopérative de textes ou de dessins en temps réel. L'intégrité a été maintenue en bloquant l'accès de portions de base de données jusqu'à la fin de leurs mise à jour. De tel blocages ne toujours conviennent pas pour une interface de informatique de groupe. Des solutions nouvelles sont nécessaires pour résoudre ces conflits. Les architectures client-serveur fonctionnent bien pour gérer des processus distribués. Mais des architectures répliques, avec leurs fragilité et leurs problèmes de synchronisation, sont nécessaires aux produits groupware afin de préserver les temps de réponse locaux acceptables. L'addition de multi-média complique encore le problème. Le contrôle de la vidéo analogique demande un environement client-serveur qui crée des délais intolérables lorsque les distances entre interlocuteurs augmentent. Cet article présente des critères de conception de informatique de groupe d'un point de vue utilisateur et fait apparaître des problèmes architecturaux profonds qui doivent être résolus si l'on veut permettre aux environements coopératifs multi-média de fonctionner de manière efficace.

Introduction

The 1980's were the decade of the personal workstation. The 2000's will be the decade of computer-supported cooperative work (CSCW), in which shared windows will open on multiple workstations across hallways and even across continents (Baecker, 1991). Video images of co-workers in both real and delayed time will complement the shared windows. Shared work environments and multi-gigabyte fiber transmissions will be commonplace. Audio connections are moving to the airwaves (cellular telephones) while more intensive data transfer (video conferencing and shared work) is taking over the high bandwidth networks (Dertouzos, 1991). As exciting and imminent as these changes are, implementing CSCW environments requires an essential rethinking of workstation hardware, distributed systems, networks and data transfer to support the new environments.

People are real time systems. Much of today's distributed processing solutions have been built on the assumptions that delays do not matter and that packet switching and asynchronous tranfer of information is acceptable (Tanenbaum, 1992). When the distributed human processor is added to the system, these assumptions are no longer valid. With individual interfaces, certain tolerances of delays and non-synchronicity of events were accepted because the work being accomplished was private. In cooperative environments, work is performed jointly and interaction requires each individual to exhibit a range of subtle behaviours for negotiation, teambuilding, expressing doubt, etc. What worked before won't work now: the social cost is far too great.

The above paragraph characterizes the nature of the constraints facing the builders of cooperative work environments. The constraints are what I call "deep" system problems, i.e., their solutions are based on the very architecture and hardware on which the system is built. The constraints are also human interface problems, i.e., they are brought on by the needs of the human user.

A common misconception in computer science is that the user interface design can be separated from the working part of a system. This misconception has led to the development of user interface toolkits and user interface management systems for supporting the design process. Although these tools have been effective in making some aspects of designing interfaces easier, they can blind the designer from perceiving more functionally based difficulties with the interface design. In CSCW, it is clear that these tools are not enough and that more fundamental changes to the underlying system are necessary if the interface is to work. Thus, the basic premise of this paper is that CSCW is one human interface area that needs to reach deep into the computer science field for support. Tools for manipulating screen objects and handling input devices no longer suffice.

Why is the design of cooperative work interfaces so challenging? Humans are social animals and have very finely tuned skills for handling social contact. If the tool they use affects their ability to project themselves or to maneuver comfortably in the environment created by the tools, the entire tool will be rejected. In contrast to individual interfaces, where the very privacy of working at the interface protects the person learning the interface from ridicule, the public nature of shared interfaces makes users extraordinarily sensitive to small problems.

The sections which follow in this paper take a human-centred point of view in examining the design issues in building CSCW systems. Human needs are stated and explained. What these needs translate to in terms of underlying system design is then discussed along with a presentation of various solutions that have been tried and the problems that have been encountered. A major difficulty with constructing an architecture that fulfills one human need of a CSCW system is that the architecture then violates another need. Often all current design solutions are found inadequate and in need of new research to make the cooperative system work appropriately.

A major focus throughout the discussion is the assumption that thousands of users will someday be working with the system so that scalability, flexibility, robustness, and support for workstation heterogeneity are important issues (Arango, et al., 1992). Many solutions for today's prototype systems, which manage 15 to 100 connections, are not viable for interoffice connections of most major corporations or government organizations which may involve thousands of simultaneous connections.

For my discussion I select two representative CSCW technologies. I begin with desktop video conferencing systems (also called media spaces), which represent those technologies designed to allow people to meet visually while still remaining a long distance from each other. This technology is invariably connected to computing technology which supports the video and audio connections but also provides some form of shared software. The interface needs for supporting shared software are treated in the next section.

There are many aspects of CSCW that I do not discuss. They include such items as shared calendars, hypertext systems, office coordination systems, video mail, group decision support systems, etc. Each of these applications is either asynchronous - and therefore does not make the demands on the distributed processing that synchronous linkage entails - or poses problems that are the same as those arising from the desktop video conferencing and shared software.

This paper is a discussion of problems not solutions, but it is one with a hopeful note. It provides a new set of constraints that help to focus the design process and suggests rethinking the mechanisms that are used to handle distributed processing and video network traffic. It suggests that solutions can be found and often points to potential paths to take especially in the real time systems area.

Desktop Video Conferencing

Desktop video conferencing involves the interconnection of offices which can be relatively far apart (e.g., in different cities) by a complete video and audio hookup (Watabe, et al., 1990; Crowley, et.al., 1990). It has some similarities to a videophone, in which the two parties in a long-distance conversation have a visual picture of each other in addition to audio. It is also more than a videophone because it supports more services, e.g., meetings, browsing, shared common areas, etc. Desktop video conferencing systems are designed to support the ubiquitous visual contact that facilitates collaboration in day-to-day communication (Kraut, Egido & Galegher, 1988). History has demonstrated a very poor market for videophone calls and video conferencing (Egido, 1988), but current research with media spaces (Goodman & Abel, 1986; Buxton & Moran, 1990) has indicated effective casual communication uses that the visual channel supports. Figure 1 illustrates a typical configuration of the type of network infrastructure that supports desktop video conferencing.

Today's desktop video conferencing architectures assume different channels of communication for the different media. The video is shipped via coaxial cable or fiber and the audio is sent either by normal telephone connections or by separately mixed audio. The computer controls switch boxes which make the connections. Tomorrow's A/V communication will be digital but still shipped by physical connections, e.g., dark fiber. This makes the soft switching of the connections an easier problem, but each type of communication must still be handled uniquely (Vin et al., 1991). This is inherent in the nature of the information being shipped. Audio for multi-person contacts needs to be mixed separately to filter out noise from each site. Video requires compression to ship. Effective compression strategies are hybrids of techniques

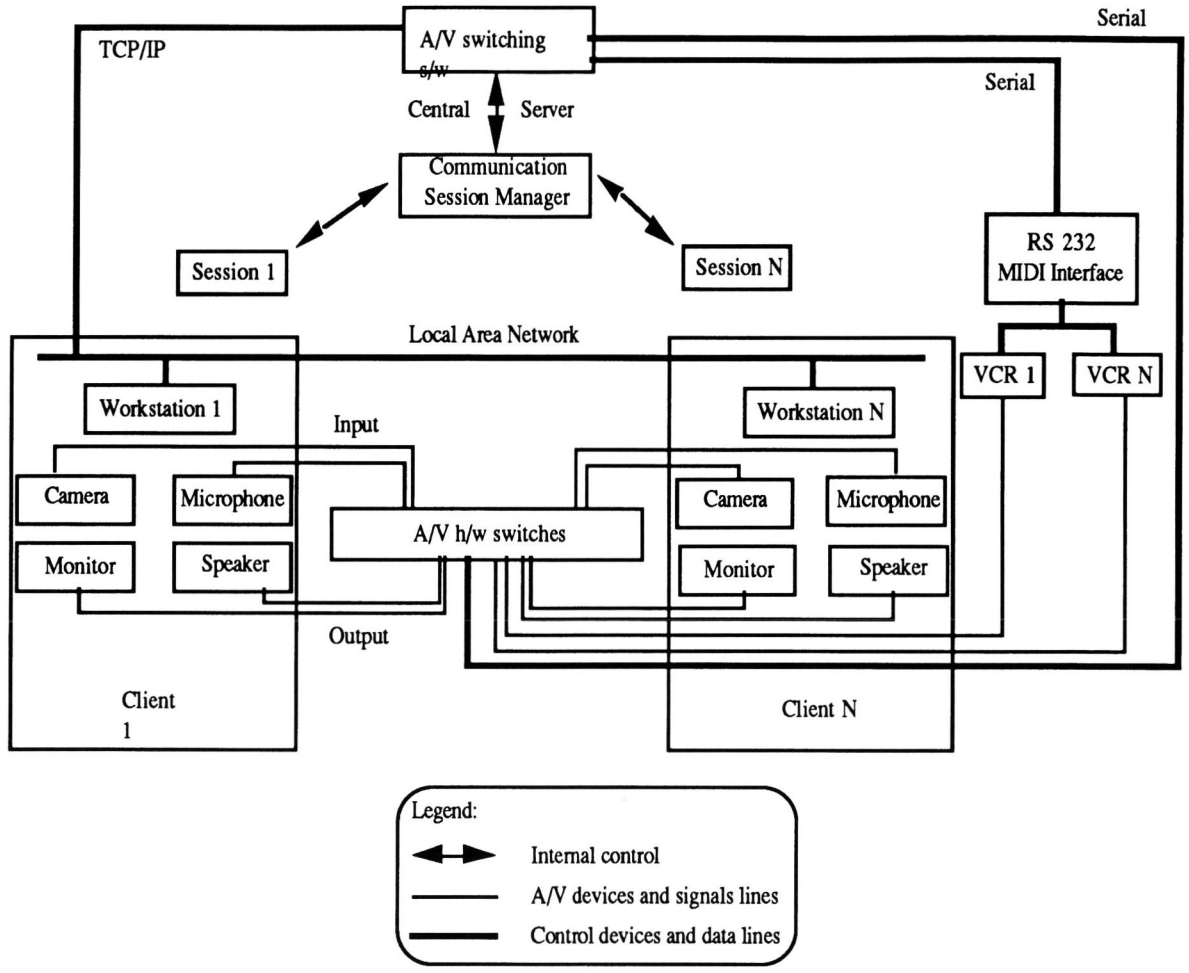

Figure 1 Typical configuration of current desktop video conferencing configurations. The video and audio are shipped by analog and a local area network passes the control informaton to a central server which is manageing the switching..

based on changes in the visual scene over time and information present in the visual scene (Fox, 1991). Thus,scenes from the different sites in the multi-person conversation will be compressed and managed differently. Exchanges of computer messages handling the shared workspaces will form the third channel. In addition to managing shared computer workspaces, the computer signals on this channel will also be used to manage the transmission of the audio and video channels, e.g, authorizing an additional person to enter into a video meeting.

Current desktop video conferencing installations ship the audio and video over analog channels and use a client-server architecture to manage the switching (Buxton & Moran, 1990; Arango et al, 1992). The client computer is located close to the switching centre and maintains a database of connection states to manage the A/V switching. Digitized video and audio transmission need not be controlled by a centralized switching arrangement opening up other possible control architectures.

In the discussions of the user interface to desktop video conferencing which follow, digitization of all signals is assumed. This opens the possibility of comparing client-server versus distributed A/V controls from the standpoint of user needs for this environment. This also opens up the issue of packaging the different varieties of digitized information that are to shipped plus a much larger issue, that of shipping the very large amounts of data arising from digital video. Current network solutions have not been designed for this type of traffic. The list of user issues associated with video conferencing is endless. Below, I discuss six important ones to illustrate the major impact of network architecture on the desktop video conferencing interface.

Modality Synchronicity

Humans communicate by many modes. They talk, they gesture, they point, they look at things with their eyes. What they are relatively unaware of is how synchronized these activities are. Gestures end at the end of a clause, eyes change direction at sentence end. Not having this synchronicity is symptomatic of underlying brain disorders. The problem is a variation of motor aphasia, and

it is very disturbing for others to engage in communication with a motor aphasic person. Despite a feeling of deep uneasiness, the recipient of the asynchronous communication cannot describe what is wrong with the conversation. An example that most of us are familiar with is the TV character, Max Headroom.

When voice and video signals are captured and shipped separately, it is likely that they will arrive slightly out of synch. Such loss of synchronization is worse than a badly dubbed movie because it destroys all the redundant cues the listener relies on for understanding the conversation. If, in addition, the individual is heard to be typing or seen to be drawing in the video picture, and the screen update information arrives late or early, the flow of discussion proceeds haltingly as each person waits for corroboration that all information has arrived. Early video conferencing systems often used a speaker phone to send the audio portion of the signal, but quickly found this setup to be inadequate. Human speech is highly overlapping (Buxton & Sellen, 1992), but speaker phone technology does not permit this overlapping. Without the visual channel, attempts to speak are not apparent. With the visual channel, the limitations of the speaker phone are all too visible.

Desktop video conferencing systems can be designed to pack and ship all signals as one packet, but this leads to other problems. Audio is very different from video and demands different handling characteristics. Although both are continuous signals, the sampling rates can be very different. Screen events will also have different closure times than either video or audio signals. A series of mouse selections taking 500 milliseconds or more may occur before it is possible to send a valid user event. Whether the audio, video and user event information is sent digitally down a single channel to be unpacked by the receiving workstation or down separate channels, synchronous display of all channels of information is essential for the success of the system.

Entering and Leaving Sessions

Unlike telephone contact, visual contact is rarely restricted to two people in an active workplace. Sighting other individuals working together in an office is often an invitation for a third and possibly fourth person to join the conversation (Root, 1988). Video sessions need to support similar behaviour. This implies a video conferencing capability in which participants can scan all video conversations and ask to join those which interest them. It also implies an environment in which people can leave a video conversation and new people can join maintaining a continuous thread of connected people even after both originators of the conversation have left.

A variety of mechanisms exist for handling multiple individuals in a video meeting. Most arrangements mix the video signal at a centralized source and ship the merged version down a single video channel to each recipient. Compressing the video signal makes mixing more difficult. Nevertheless, this can be done at a communications bridge which unpacks each signal, mixes it and then compresses it again for shipping. An inherent disadvantage is incurred in having to unpack the video signal twice. Mixing the video at a centralized site makes it easier to build interfaces that allow others to peer into meetings that are going on but centralized mixing is an architecture that does not scale up very well. Centralized video mixing also means that a software bridge is needed for each meeting that is taking place, even for two person meetings because of the potential for adding more participants to these meetings.

Video signals can alternatively be sent directly to each workstation and mixed there, but this implies that each workstation have the ability to handle that level of input traffic. Since digitized video signals already require high bandwidth to ship, this is currently not a feasible arrangement.

Figure 2 A user is seated in front of three Hydra units. In the photo, the Hydra units sit on the table in front of the chairs that would otherwise be occupied by three remote participants. Each Hydra unit contains a video monitor, camera, and loudspeaker. A single microphone conveys audio to the remote participants. (From Buxton & Sellen, 1992)

Hydra is an example of video meetings that are not mixed at a central site (Buxton, 1991). It is one of the more innovative interfaces that solves the problem of audio and positional location in video conferences. Hydra provides a video image, camera, speaker and microphone in a small desktop unit for each person engaged in the meeting. If four people are meeting, a Hydra user would have three units on the desktop, each one representing a virtual person in the meeting. Figure 2 contains a diagram of one of the Hydra units. Hydra currently works by shipping video signals down separate channels. This type of transmission uses an enormous bandwidth. Hydra also has no mechanism for displaying a visual picture of the meeting taking place to others who might want to join the meeting. Although Hydra solves important social considerations for meetings,

its implementation in a large scale communication system would be very difficult

Maintaining Continuous Sessions

Our research at the University of Toronto has shown that one of the very viable modes of a desktop video conferencing environment is an open channel (Li & Mantei, 1992). A visual channel is maintained continuously between one or more offices to support ad hoc information exchange by people working closely together. These connections are maintainable with the architecture shown in Figure 1 but do not handle a private video call. A private call is different from a meeting. The caller wants to talk to one person only. Such a setup is equivalent to having a two-party line, but the line needs to support the visual contact for both parties yet not connect all other parties into a video conference. The setup also needs the capability of connecting the parties into a video conference, if at some point, the caller wants to join the larger group. Thus, the underlying connection architecture needs to be able to switch the caller to the software bridge which is maintaining the other connections

Privacy Considerations

Up until now, we have only discussed the problems of making different multiple connections and have not addressed the issue of preventing users of the system from making visual contact. It is believed that being able to

view co-workers at work provides useful opportunities for communication which co-workers take advantage of (Kraut et al., 1988). Limiting the video channel to a meeting or a video call prevents these visual opportunities yet keeping an open visual channel infringes on individual privacy. What is needed is a system that is both designed to permit large amounts of visual browsing but that also allows participants to limit browsing intrusions by giving browsing rights to a subset of people. The underlying design issue is where to keep the privacy informaiton, locally or in a centralized database.

Privacy settings can be kept at each desktop node, but if people are browsing meetings, then the privacy considerations of each person in the meeting need to be handled. Such privacy screening can cause long delays for larger meetings where nodes in many different cities require checking. A faster way is to manage meetings at a centralized location with privacy being the lowest common denominator of the assembled group.

Soft Communication Failures

A common problem with long distance communication technology is one of providing alternatives to communication failures. The basic failure is one of not reaching the individual called. Voice mail and the telephone answering machine are solutions to these problems. When direct contact fails, there is a backup storage device for leaving an audio message. This works well for voice communications, but video contact because of its much larger bandwidth and cost of storage may not be amenable to the same solutions. Currently, voice mail storage is centralized. Callers record their message and have options of replaying it and re-recording it if it sounds unsatisfactory. Users can then call the central storage and access their calls. This setup works reasonably well for voice messaging. How would it work for video messaging?

A caller would send a video message to a centralized site. This would be digital video involving a very large amount of storage. If the caller wanted to play back the message, the digital video would be sent back to the caller. This is a large volume of network traffic which can be avoided by local playback and shipping when the message is completed. However, local playback requires local storage which is currently expensive for digital video. This problem can be cheaply bypassed by storing analog video which can be digitized when the message is finally ready to transmit. Local video storage is also prone to failure which implies additional backup strategies that use (1) other local storage devices or (2) a centralized device when all else fails. The underlying argument is that mechanisms for supporting transitions to video mail cannot be handled by using the current storage solutions for telephony.

Support for Multiple Contacts

Video wears two hats in the office. It is a broadcast medium and a meeting tool. This implies support for both functions in the desktop video conferencing environment. It also implies the capability to move between either function with little difficulty. Broadcast video involves a multipoint setup that distributes a single video image to many sites but does not necessarily receive video images from these sites. Broadcast connections are often sent to interrupt other connections such as the audio landing notifications on airplanes which interrupt music channels, but broadcasts can often be secondary information that users would like to see interruptible by video phone calls. A network which manages broadcasts is configured differently than a network which handles point to point contacts. In a desktop video conferencing environment, both activities need support.

Activity Sharing Software

Activity sharing applications are programs in which multiple individuals working on separate workstations can simultaneously be viewing and updating the same work. These workstations can be a few metres or hundreds of kilometres apart. Messages sent along networks joining the workstations update the screens with each users input. A variety of such programs exist for different workstation platforms and some are commercially available such as Aspects[tm] (Group Technologies, 1991). Shared activity applications primarily support shared writing and shared drawing tasks (Fish, et al., 1988; Tang & Minneman, 1990, Greenberg & Bohnet, 1990), but prototypes also exist for supporting programming (Brothers, Sembugamoorthy & Muller, 1990) and database design work (Hayne & Ram, 1990). Most software has been written as application specific, e.g., for wordprocessing only, but some work has been done to build an underlying operating system or window system that will support a wide variety of single user applications (Lauwers & Lantz, 1990; Lauwers et al., 1990).

Two main types of architecture support the shared activity sessions. The first is a client-server architecture in which the application resides on a designated workstation and all other workstations send messages to update the data structure residing in the application (University of Michigan, 1990). The application, in turn, sends out messages to update the screens on each workstation. The second is a replicated architecture. (Replicated architectures are also called serverless or peer-to-peer architectures.) In the replicated architecture, an application resides on each workstation and exchanges messages with all other workstations, accepting input generated by other users and sending input to update the screens of the other users (Crowley et al., 1990). Replicated architectures have serious fragility problems (Ahuja, Ensor & Lucco, 1990) because of difficulties with synchronization and heterogeneity in workstations whereas client-server architecture generate a large amount of message traffic and have to maintain states of all workstations. Hybrids of partially replicated and part client-server architectures also exist (Mawby, 1991; Lu, 1992) which bring the advantages and disadvantages of both solutions.

From the human perspective, either solution has problems in supporting collaboration. Instead of either extreme, the hybrid between the client-server and replicated architecture needs to be tuned very carefully to meet the user constraints listed in the sections that follow. The shared architectures need to be concerned with maintaining adequate system response time, adding latecomers to a work session, not disrupting the flow of work with lockout procedures, synchronizing the work between collaborators, etc.

Response Time

A user who is developing drawings using a mouse or a stylus interface needs to have the immediate feedback on the results of their motor behaviour displayed on the screen if they are to use their finely tuned hand-eye coordination capabilities. Lags in displaying figures on the screen are unacceptable and make the drawing task extraordinarily difficult. Current client-server arrangements which send the user input to the server and then back to the client are far too slow for this microsecond coordination.

Users also need to see the drawing behaviour of their collaborator in real time. If their collaborator is drawing a box or circle or selecting text to delete, the size and placement of of the figure or the span of text selected show up on the screen as feedback to the user before a key is released or other action taken to indicate acceptance of the sizing, placement or text selection outlined on the screen. If these events are not shown in real time to the collaborator it is difficult for the collaborator to comment on the work being done (Lu & Mantei, 1992). A client-server architecture could synchronize these times better than replicated architectures especially if distances between sites were large, but both architectures will have problems with synchronizing this continuous event traffic. This is a real time concern.

Hand-eye coordination is important to support in shared wordprocessors as well as shared drawing tools. Text that is delineated by a mouse controlled cursor is a variation of drawing behaviour. Gesturing and pointing on text require similar hand-eye coordination and thus, real time support.

Adding Latecomers

Meetings rarely begin or end with all people present. Attendees join sessions late and others leave early. This requires updating the newcomer to what has transpired in the meeting and making sure that the early leaver has a final document of all the decisions made at a meeting.

In a shared activity session, the problem of updating the workstation of a latecomer is a different task depending on whether a client-server or a replicated architecture is used for managing the shared session (Chung, 1991). With a client-server architecture, the problem is one of setting the window parameters of the newcomer and updating the shared session information of the other participants. With a replicated architecture, the problem is one of transferring a copy of the work to the newcomer's workstation. On the social behaviour side of the issue, the problem is one of integrating the newcomer into the meeting with little disruption. The newcomer needs to obtain a copy of the current state without interrupting the flow of changes and additions that are being made to the work product. Currently, most replicated architectures lock out all work until the newcomer's workstation is updated. Others simply do not permit latecomers. Client-server architectures such as Rendezvous (Patterson et al., 1990) can support a more graceful entry but run into problems in providing an accurate replication of other user's workstation environments. This problem becomes particularly acute in heterogenous workstation environments. Although the client-server architecture is much better at handling the latecomer problem, the effects of a client-server architecture on response time latency strongly support a replicated solution.

Lockout and Concurrency Issues

Up until recently, users have not had the capability to work in the same space at the same time. Shared writing, drawing and spreadsheet software give us this functionality but not without integrity concerns. If one user deletes a sentence at the same time another user is modifying it, what is the end result? One solution is to prevent modification access to any object that is already in use by another user (Ellis & Gibbs, 1988), but this creates new problems. To be locked out, text or figures need to inform all workstations of their modification status. Both the informing process and the interrogation process take time. Users neither like waiting for access to a screen object (e.g., a word) before they can use it or being denied access seconds after they have attempted to modify the screen object. The denial of access is a particularly acute problem because users have already begun their cognitive work plan when they attempt to select the object (Card, Moran & Newell, 1983). Users want to know in advance when text or figures are locked out to avoid unproductive mental effort.

In most writing tasks, concurrency is not an issue. Lockout is usually at the character level as is seen in such shared editor systems as Aspects (Group Technologies, 1991), ShrEdit (University of Michigan, 1990), and

SASE.(Mawby, 1991). With such a fine resolution for lockout, the probability of concurrent access is low. Writing is such a complex cognitive process, that people composing text prefer to do it on their own (Posner & Baecker, 1991). However, one of the major times when users are likely to collide occurs frequently. When one participant in a shared session is typing an idea and others are observing its generation, typing or spelling errors occur which other users jump in to fix. They cannot perform these correction because this text is locked during creation.

Ellis, Gibbs and Rein (1988) give a good presentation of alternate solutions to lockout which support a more flexible participant environment. One of their suggestions is that of allowing individual work to diverge and putting the onus of the repair on the participants. Another possibility, especially for drawing tasks, is to ignore concurrency and to throw away messages that request changes to objects that no longer exist. This is done in CaveDraw (Lu, 1992) where one user can delete the very layer another user is drawing on. Collision events are assumed to be relatively infrequent and negotiable by the other multi-media support available. For other shared software tasks, similar optional concurrency control measures can be in place. This implies that the software which controls concurrency issues will need to be smarter and know when and where concurrency will be problematic.

View Synchronicity

Users of collaborative tools often do not use them in a collaborative fashion. There are collaborative tasks that people perform that are so complex, e.g., writing or programming, that interacting with others inhibits the performance of the task (Neuwirth, 1990). Often collaborative work sessions are a combination of people working together, parceling out the work to be done and then working individually on the work. The advantage of a collaborative work environment is that it allows co-workers to smoothly move into and out of collaborative mode. This type of observed movement between collaborative and non-collaborative states is common in computer supported meeting rooms (Mantei, 1988).

The underlying application support for allowing both types of work patterns in a collaborating group will permit participants to have different views of the work product. It also will have a functionality for synchronizing participant's views. Synchronized views work in tandem. When one participant scrolls to a new place in the text, the other participants see their windows scroll as well. To synchronize views, the software needs to maintain information on the screen states of each of the participating workstations. Since individual users may have differing window sizes, use different fonts and even have different objects occluded, synchronization requires a large amount of mapping operations between each workstation.

Since a sequence of user events at one workstation is often necessary to determine the synchronization at the other workstation, arbitrarily breaking up the events into message packets can destroy this context. For example, a user can scroll to a position and then select the visible text. The workstation that is synchronized with the first workstation can scroll to an approximate location, but if text is selected in a part of the window on the first workstation that is occluded on the second workstation, the second workstation will have to execute a second scroll and then show the text selection. This type of synchronization looks jumpy and awkward to the participants, not to mention the concern that both participants are still not seeing the same text.

Occlusion in many window systems may be difficult to detect because one of the fundamental premises of window systems is that applications should be able to write into windows without concern for what portions of the window are occluded (Scheifler, Gettys & Newman, 1988). Without control of occlusion, view synchronization becomes a difficult problem.

Synchronization is also one case where concurrency control is necessary. Once screens are synchronized, scroll bar usage can lead to "scroll wars" as can other global events which update the screen (Stefik et al., 1987). So, although synchronization is very much at the interface level, mechanisms that pass the streams of user behaviour between the synchronized workstations also have to examine these event streams and either require homogeneity in the resources used by the participants or interpret the streams to best represent a synchronized environment.

Gesture Support

When people work together, they use their hands to point, circle,or motion in a wide variety of ways about the information that is being created (Tang, 1989). Support for gesturing in shared activity software comes in the form of a telepointer, that is, a cursor which is seen on everyone's screen that is controlled by the owner of the telepointer. Each participant in a shared work session has a personal telepointer, usually uniquely identified by shape or colour. Each participant also has their own cursor for moving around their workscreen.

Workstations therefore need to support multiple cursors, i.e., the cursor of the person at the workstation and the cursors of any of the participants that are in telepointer mode. Since cursor functioning is done at low levels in systems software and since most workstations are rarely equipped to handle more than one mouse input, this support suggests fundamental changes in workstation and/or operating system design. Workstations will need to support multiple cursors, and if necessary, to change the shape and form of the cursors as they move over the windows (Greenberg, 1991).

Version Control

In a replicated architecture, which workstation has the most recent version? Is it the one that left the session last, the one with the latest time stamp, or the one that has been designated to hold the latest version? If an individual works alone on the work product after the session, is the update automatically transferred to each of the other

workstations? What happens if two people work on the file separately but at the same time? Storage mechanisms need to be put in place that allow multiple users to maintain version control of the work product. This is not a hard technological problem. What is necessary is that the solution be incorporated in the basic architecture of the system rather than at the application level. Otherwise the latest version might reside on a workstation that is not in operation or connected to the next joint work session. Humans are notoriously bad at version control. It is best to leave this task to the system.

Conclusion

In this paper I have discussed a set of requirements that need to be met if the social nuances of collaboration are to be supported by cooperative work tools. What is important about these requirements is that the ability to meet them lies in solutions at the very heart of the underlying system architectures and communication structures that support CSCW. Small adjustments at the interface level will not fix the problems. In some cases, only redesign of the computer workstation and/or its operating system will make the problem solutions doable Some of the most viable fixes for one requirement contradict the most viable solution for another. Some of the solutions violate current ways of thinking about the world, i.e., we don't need to guard against concurrency. I haven't presented solutions to these problems. I am not the systems guru. However, demonstrating a problem's existence is always the first step in its solution.

Acknowledgements

For research support the author is indebted to the Natural Sciences and Engineering Council of Canada (NSERC), the Information Technology Research Centre of Ontario (ITRC), Apple Computer, Inc., Rank Xerox EuroPARC, Digital Equipment Corporation, IBM Canada's Laboratory Centre for Advanced Studies and Alias Research. I also wish to thank Catherine Plaisant for translating the abstract into French, Kevin Schlueter for his many insightful comments on the manuscript and Jin Li and Gifford Louie for their artwork.

References

Ahuja, S. R., Ensor, J. R., and Lucco, S. E. (1990) A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems. *Proceedings of COIS'90 Conference on Office Information Systems*, Cambridge, MA, April 25-27, 1990, pp. 238 - 248. New York: ACM Press.

Arango, M., Bates, P., Fish, R., Gopal, G., Griffeth, N., Herman, G., Hickey, T., Leland, W., Lowery, C., Mak, V., Patterson, J., Ruston, L., Segal, M., Vecchi, M., Weinrib, A. and Wuu, S. (1992) Touring Machine: A Software Platform for Distributed Multimedia Applications, *Proceedings of 1992 IFIP International Conference on Upper Layer Protocols, Architectures and Applications*, Vancouver, BC, Canada, May 1992.

Baecker, R. (1991) New Paradigms for Computing in the Nineties. *Proceedings of Graphics Interface'91*. Calgary, AB, Canada, June 1991, pp. 224 - 229.

Brothers, L., Sembugamoorthy, V. and Muller, M. (1990) ICICLE: Groupware for Code Inspection. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 169 - 181, New York: ACM Press.

Buxton, W. A. (1991). Telepresence: Integrating Shared Task and Personal Spaces. *Proceedings of Groupware'91*, Amsterdam, Holland, October 1991.

Buxton, W. A. and Moran, T. P. (1990) EuroPARC's Integrated Interactive Intermedia Facility (iiif): Early Experience. *Proceedings of the IFIP WG 8.4 Conference on Multi-user Interfaces and Applications*, Heraklion, Crete, pp. 11 - 34. In S.Gibbs and A. A. Verrijin,-Stuart (Eds.) *Multi-User Interfaces and Applications*. Amsterdam: North-Holland.

Buxton, W. A. and Sellen, A. J. (1992). Interfaces for Multiparty Video Conferences. *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, Monterey, CA, May 3-5, 1992. New York: ACM Press.

Card, S. K., Moran, T. P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Chung, G. (1991). *Accomodating Latecomers in a System for Synchronous Collaboration*. Unpublished masters dissertation, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, August 1991. Available as report no. TR91-038.

Crowley, T., Milazzo, P., Baker, E., Forsdick, H. and Tomlinson, R. (1990). MMConf: An Infrastructure for Building Shared Multimedia Applications. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 329 - 342, New York: ACM Press.

Dertouzos, M. L. (1991). Communications, Computers and Networks. *Scientific American* 265(3), September 1991, pp. 62 - 69.

Egido, C. (1990). Video Conferencing as a Technology to Support Group Work: A Review of its Failures. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 13 - 25, New York: ACM Press.

Ellis, C. A., Gibbs, S. J. and Rein, G. L. (1988). *Groupware: The Research and Development Issues*. MCC Technical Report Number STP-414-88, Austin, TX.

Ellis, C. A. and Gibbs, S. J. (1988). *Concurrency Control in Groupware Systems*. MCC Technical Report Number STP-417-88, Austin, TX.

Fish, R., Kraut, R., Leland, M. and Cohen, M. (1988). Quilt: A Collaborative Tool for Cooperative Writing. *Proceedings of COIS'89 Conference on Office Information Systems*, Palo Alto, CA, March 23-25, 1988, pp. 30 - 37. New York: ACM Press.

Fox, E. A. (1991). Advances in Interactive Digital Multimedia Systems. *IEEE Computer* 24(10), October 1991, pp. 9 - 21.

Goodman, G. and Abel, M. (1986) Collaboration Research in SCL. *Proceedings of CSCW'86 Conference on Computer Supported Cooperative Work* , Austin, TX, December 3 - 5, 1990, pp. 246 - 252, New York: ACM Press.

Greenberg, S. (1990). Sharing Views and Interactions with Single-User Applications. *Proceedings of COIS'91 Conference on Office Information Systems,* Cambridge, MA, April 25-27, 1990, pp. 227-237. New York: ACM Press.

Greenberg, S. and Bohnet, R. (1991). GroupSketch: A Multi-User Sketchpad for Geographically Distributed Small Groups. *Proceedings of Graphics Interface'91*, Calgary, Alberta, June 5 - 7, 1991.

Group Technologies (1991). *Aspects: The First Simultaneous Conference Software for the Macintosh, Version 1*. Manual, Group Technologies, Inc., Arlington, VA.

Hayne, S. and Ram, S. (1990) Multi-user View Integration System: An Expert System for View Integration. *Proceedings of the IEEE International Conference on Data Engineering*, Los Angeles, CA, pp. 402-409.

Kraut, R., Egido, C. and Galegher, J. (1988). Patterns of Contact and Communication in Scientific Research Collaboration. *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 26-28, 1988, pp. 1 - 12. New York: ACM Press.

Lantz, K. A. (1986). An Experiment in Integrated Multimedia Conferencing. *Proceedings of CSCW'86 Conference on Computer-Supported Cooperative Work*, Austin, TX, December 1986, pp. 267 - 275. New York: ACM Press.

Lauwers, J. C. and Lantz, K. A. (1990). Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems. *Proceedings of CHI'90 Conference on Human Factors in Computing Systems,* Seattle, WA, April 1990, pp. 303 - 312. New York: ACM Press.

Lauwers, J. C., Joseph, T. A., Lantz, K. A. and Romanow, A. L. (1990). Replicated Architectures for Shared Window Systems: A Critique. *Proceedings of COIS'90 Conference on Office Information Systems*, Cambridge, MA, April 25-27, 1990, pp. 249 - 260. New York: ACM Press.

Li, J. and Mantei, M. (1992). Working Together, Virtually. *Proceedings of Graphics Interface, 92*, Vancouver, BC, Canada, May 13-15, 1992.

Lu, I. M. (1992). *Supporting Idea Management in a Shared Drawing Tool*. Unpublished Masters dissertation, Department of Computer Science, University of Toronto, Toronto, ON, Canada, January 1992.

Lu, I. and Mantei, M. (1992). *Managing Design Ideas with a Shared Drawing Tool*. Unpublished Manuscript, Department of Computer Science, University of Toronto, January, 1992.

Mantei, M. M., Baecker, R. M., Sellen, A.J., Buxton, W. A., Milligan, T. and Wellman, B. (1991). Experiences in the Use of a Media Space. *Proceedings of CHI'91 Donference on Human Factors in Computing Systems,* New Orleans, LA, April 28 - May 4, 1991, pp. 203 - 208. New York: ACM Press.

Mantei, M. (1988). Capturing the Capture Lab Concepts: A Case Study in the Design of Computer Supported Meeting Environments. *Proceedings of CSCW'88 Conference on Computer Supported Cooperative Work*, Portland, OR, September 26-28, 1988, pp. 257 - 270. New York: ACM Press.

Mawby, K. L. (1991). *Designing Collaborative Writing Tools*. Unpublished Masters dissertation, Department of Computer Science, University of Toronto, Toronto, ON, Canada, September 1991.

Neuwirth, C. M., Kaufer, D. S., Chandhok, R. and Morris, J. H. (1990). Issues in the Design of Computer Support for Co-authoring and Commenting. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 183 - 195. New York: ACM Press.

Patterson, J. F., Hill, R. D., Rohall, S. L. and Meeks, W. S. (1990). Rendezvous: An Architecture for Synchronous Multi-user Applications. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 317 - 328. New York: ACM Press.

Posner, I. R. and Baecker, R. M. (1991). How People Write Together. *Proceedings of the 25th Annual Hawaii International Conference on Systems Science, Vol IV* Kauai, HI, January 7 - 10, 1992, pp. 127 - 138.

Root, R. W. Design of a Multi-Media Vehicle for Social Browsing. *Proceedings of CSCW'88 Conference on Computer Supported Cooperative Work* , Portland, OR, September 26 - 28, 1988, pp. 25 - 38. New York: ACM Press.

Scheifler, R. W., Gettys, J. and Newman, R. (1988) *X Window System*. Burlington, MA: Digital Press.

Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S. and Suchman, L. (1987). Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM* 30(1), January 1987, pp 32 - 47.

Tang, J. C. (1989) Listing, Drawing and Gesturing in Design: A Study of the Use of Shared Workspaces by Design Teams. Unpublished Ph.D. Dissertation, Stanford University, April 1989. Also available as*Xerox Technical Report SSL-89-3*., Xerox PARC, Palo Alto, CA.

Tang, J. C. and Minneman, S. L. (1990). VideoDraw: A Video Interface for Collaborative Drawing. *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, Seattle, WA, April 1990, pp. 313 - 320. New York: ACM Press.

University of Michigan. (1990) ShrEdit 1.0: A Shared Editor for the Apple Macintosh -- User's Guide and Technical Description, June 1990. Cognitive Science and Machine Intelligence Laboratory, University of Michigan, Ann Arbor, MI.

Vin, H. M., Zellweger, P. T., Swinehart, D. C. and Rangan, P.V. (1991). Multimedia Conferencing in the Etherphone Environment. *IEEE Computer* 24(10), October 1991, pp. 69 - 79.

Watabe, K., Sakata, S., Maeno, K., Fukuoka, H. and Ohmori, T. (1990). Distributed Multiparty Desktop Conferencing System: MERMAID. *Proceedings of CSCW'90 Conference on Computer Supported Cooperative Work* , Los Angeles, CA, October 7 - 10, 1990, pp. 27 - 38. New York: ACM Press.