

A Survey of Collaborative Systems

1. INTRODUCTION

Remote multi-user interaction in a collaborative environment is one of the hottest topics today in Distributed Computing, Web Applications and Tele-learning. Their application area ranges from games to Tele-Laboratories, Tele-Medicine, Network Management, Tele-Commerce and others.

Tele-conferencing and video-conferencing support only one part of the collaborative process, that of bringing people together.

Yet most real meetings require not only the people, but also the materials and on-going work participants wish to share with others. These include notes, documents, plans and drawings, as well as some common work surface that allows each person to annotate, draw, record and convey ideas during the meeting's progress.

Given that an individual's work is commonly centered around a workstation, the networked computer can become a valuable medium for collaborators to share work with each other, and particularly the wireless computing devices, where the mobility of them also opens up the potential for inter-group collaboration.

2. CLASSIFICATION OF COLLABORATIVE SYSTEMS

People interact together in all aspects of life and, as computers have become prevalent, users seek computer support for their interactions. Collaborative interactions fall in a range from *asynchronous*, where the points of interaction are separated by relatively long periods of time, to *synchronous*, where the interactions are simultaneous or separated by short periods of time (cf. [15]).

For providing synchronous collaboration there are different approaches.

2.1 User perspective: Collaboration Awareness VS Transparency

The ideal approach towards building software systems that support real-time sharing of work would recognize the existence of each participant and his or her niche in the collaboration.

The most effective way of accomplishing this is to develop special-purpose applications designed for simultaneous use by multiple users.

Such approach is called *collaboration awareness* (cf. [21]). The developers are *aware* that their application would be used collaboratively, so they designed collaborative features.

For example, groupware applied to document collaboration could: recognize the roles of the primary writers, reviewers and readers; adjust access permissions to reflect these roles; keep track of version differences and enhance communications between the various collaborators (cf. [22]).

Although there are collaboration-aware versions of some classes of applications, such as whiteboards and text editors, collaborative versions of others, such as a computer-aided design tools and debuggers, have yet to appear. While it is usually possible to create a collaborative version of such a system, most users do not have the required resources (i.e.

NOKIA Research Center
Giovanna Ferrari

April 2001

source code, compilers, developers). Collaboration could be built into future versions of a system, but that does not address an immediate need to work together, and many systems will never be revised to include collaboration (cf. [3]).

An example of collaborative awareness system is **GroupKit** (cf. [25]); it and its applications run on Unix workstations under a X11 environment. It supports real time distributed multipoint conferences between many users together with building of text or graphics.

The alternative approach is sharing a single-user application between participants of an on-line meeting through a "shared screen" or "shared window". Each participant would have an identical view of the running application and an opportunity to interact with it, but the application itself would have no awareness that more than one person was using it (cf. [13]).

This is the *collaboration transparency* or "sharing view". This kind of systems provide the shared use of existing single-user applications through mechanisms that are unknown, or "transparent," to the application and its developers (cf. [21]). Collaboration transparency systems can be used to collaborate in legacy applications that were developed with no support of collaboration in mind.

In contemporary window-based environments, this requirement has led to the development of *shared window systems*. Those may also allow a user to take part in a number of teleconferences simultaneously, sharing different subsets of his applications (and associated windows) with different sets of users.

Application sharing is far from new. In the 1968 Doug Engelbart built what is probably the first shared screen conferencing as part of his **NLS system** (cf. [9]). Six displays were arranged on a table so that a group of twenty participants could see the screens. While only one participant could control the screen, other participants could control a large arrow (a telepointer) visible on all screens, but invisible to the application.

Collaboration awareness and transparency offer widely different degrees of support for collaboration. Each limitation of conventional collaboration transparency contrasts directly to a capability seen in some collaboration aware applications and toolkits. Generally, well-designed collaboration aware applications provide better support for collaboration than an application sharing system, and collaboration aware systems are the only choice for inherently collaborative tasks. A collaboration aware application can provide internal support for cooperative work, such as participant properties and roles, whereas a collaboration transparency system supports collaboration externally from the application. The reason for that is because this kind of system share existing, single-user applications, were created to support one individual at a time.

However, although collaboration aware systems may appear generally superior to collaboration transparency systems, not all of them include advanced group awareness components. The omission of multi-user components from a groupware toolkit means developers must themselves implement support for desired groupware usability features.

	ADVANTAGES	DISADVANTAGES
Collaboration Transparency	Collaborate using legacy applications No extra development cost	Difficult to implement concurrent work Strict WYSIWIS Limited group awareness information Single user tasks
Collaboration Awareness	Concurrent work possible Relaxed WYSIWIS Detailed group awareness information Can support inherently collaborative tasks	Extra development cost Potential advantages not always realized

Table 1: Comparison of Conventional Collaboration Transparency versus Awareness

2.2 Architectural perspective: Centralized VS Replicated

Two architectural alternatives for constructing shared systems are the *centralized*, where the shared application is maintained in one physical location, and *replicated* approach, where the shared application is copied to each collaborator (cf. [21]). In both there is a *conference agent* that is interposed between applications and the window system.

In the *centralized* approach, or "display broadcasting" a *central conference agent* meditates all distributed work surfaces and the participants receive only a graphical depiction of the shared application (Fig1) (cf. [2]). The central agent receives all user input and merges the events so that a single instance of the shared application receives a single stream of events (cf. [18]). The central agent then distributes display updates to the participants' windowing systems.

An example is **WScrawl** (cf. [23]), a public domain group drawing program that runs on X window system. The single *WScrawl* program acts as the central agent that decides what to do with user events and where to display the output. In *WScrawl* the participant process is simply the X window server.

The advantage of that approach is that synchronization is easy, as state information is consistent since it is all located in one place. The disadvantages are that the complete system is now vulnerable to the failure of the central agent and it could be a network bottleneck, as all activity must be channeled through it.

At the other side is the *replicated* architecture, also called "event broadcasting", each participant has a copy of the application and user inputs are distributed to each copy (Fig.2). One component of a central agent here is an *event broadcaster* whose job is to multiplex an input event from any participating collaborator to every other collaborator.

For example, if one collaborator moves a scrollbar, all participants see their corresponding copy of the scrollbar move, because every collaborator invokes events to the system and those events along with the feedback that the events create are seen by all collaborators.

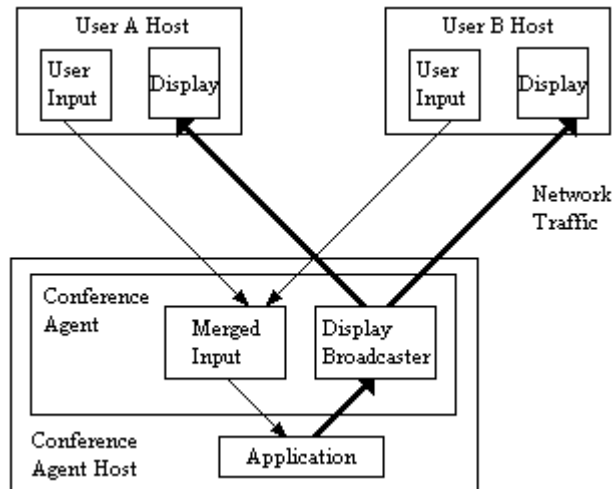


Figure 1. A centralized application shared via display broadcasting. Bold arrows indicate (high bandwidth) display data. Thin arrows indicate event traffic.

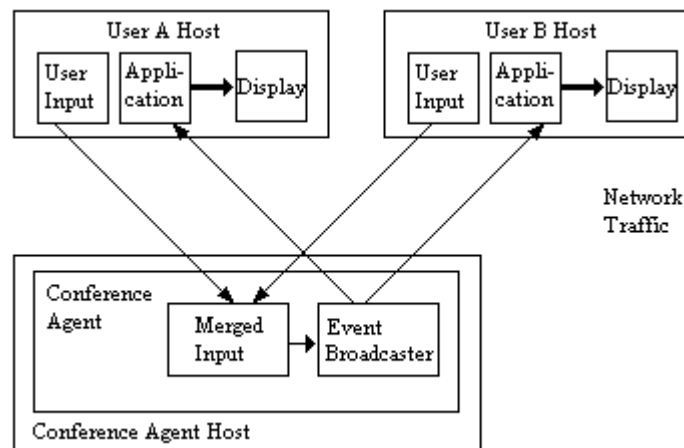


Figure 2. A replicated application shared via event broadcasting. Arrows indicate event traffic.

Replication offers many key advantages over the centralized architecture, first of all the performances. In the former the network traffic requirements are lower than in the latter, because only the participants' inputs must be transmitted, instead the centralized approach uses additional bandwidth to distribute the display information. This saving is especially relevant when sharing applications over a wide-area network, like the Internet. A replicated architecture is also more versatile than a centralized one and the system is more robust to network and machine failure. On the other hand, the replication architecture requires that all replicas of the application be synchronized all the times. In Table 2 summarizes the differences between the two approaches.

The replicated approach is implemented in a system called *Dialogo* (cf. [20]), where, although a controller's input is sent to all replicated applications, each application is responsible for generating its own output in its own workstation without going over the network.

Other examples of the replicated approach were designed by Greenberg (cf. [14]), **GroupSketch**, a multi-user sketchpad; and **GroupDraw**, a multi-user drawn package.

In the former the participant processes communicate via Unix stream sockets, but here, the only actions that can be done by a process are to either draw/erase on a bit mapped surface or to move the cursor, so there is no need to synchronize user activity. Instead **GroupDraw** is an object-oriented drawing program and the interaction between replicated participant processes is more complex. For example, to ensure that the object's behavior is managed consistently between two users, when they are grabbing the same point on a line only one succeeds. This problem here is resolved with the definition of an *owner* for each object drawn, who has ultimate authority on all operations affecting the object.

Hybrid approaches are also possible. For example, the participant processes may use a central agent only for synchronization and for mediating conflicting user requests. All other activities are performed within and between participant processes.

	ADVANTAGES	DISADVANTAGES
Centralized Architecture	Facility in providing groupware awareness Better maintenance of the synchronization	Worse response time Higher network traffic Less versatile Worse suited to accommodate heterogeneous hardware Less robust
Replicated Architecture	Better response time Lower network traffic More versatile Better suited to accommodate heterogeneous hardware More robust	Difficulty in providing groupware awareness Not easy the maintenance of the synchronization

Table 2: Comparison of Centralized versus Replicated Architecture

2.3 Groupware principles

2.3.1 Floor control

In multi-user applications, there is often the need to decide who controls what, that is, for policies of what is called *floor control*.

The policies might be classified along a number of dimensions, some of which are:

- The degree of *interaction* required. We might think of floor control as being either *automatic* (i.e., decided entirely by the application without user input) or *interactive* (i.e., decided by the user or with the assistance of user input). Interaction is required, for example, if users must explicitly “request” or “release” control.
- The extent to which user *characteristics* influence a policy. A policy might be uniform or *fair* with respect to users, or it might be user-dependent or role-dependent. For example, a *chairperson* role might allow a single user to *mediate* floor control decisions, or *priorities* associated with users might help resolve conflicting requests for control.

NOKIA Research Center
Giovanna Ferrari

April 2001

- The *granularity* of control. A policy might implement a single global thread of control for a whole application, or it might provide independent control over individual parts of an application (i.e., *objects*). I call the latter *object-specific* floor control.
- The *duration* of control. A policy might exercise control on either a *long-term* or a *short-term* basis (cf. [4]).

In some respects, particular floor control policies are similar to scheduling policies for operating systems (cf. [30]).

For example, four methods for acquiring and releasing the floor are:

Ring-passing. The participant currently in charge must explicitly release control before anyone else can assume it. Although this guarantees that a person can retain control of the floor, tension can arise between a participant who will not release control and others who wish to acquire it. Even in a benign setting, the current controller still has to remember to release the floor after he completes his turn.

Pre-emptive. Any participant may grab control of the floor at any time. This can lead to excessive interruptions in an aggressive conference.

Time slices and time outs. A person may have a set time-slice for control, after which the floor is taken away from him. More reasonably, the floor can be released for anyone's acquisition if the current controller has been idle for a set time period. Although a controller cannot be interrupted while "speaking", the floor is automatically available after a suitable pause.

Moderated. A designated participant may act as a chairperson who is responsible for passing control to and from the other attendees.

Other explicit floor control protocols include round robin, queuing, random access, and so on. The preferred style will depend upon the group task, the size of the group, the politics of the group's interactions, and the application itself (cf. [13]).

In general, the degree to which collaborators work closely together or independently is referred to as *tight* versus *loose coupling* (cf. [6]). During synchronous collaboration, floor control imposes a tightly coupled mode of collaboration because while one participant is in control, the others may only observe the work. Thus, floor control limits the collaborators' ability to work in parallel, but it is very useful to avoid potential problems associated with interleaving input events from multiple collaborators.

Collaboration transparency systems must maintain the intended behavior of the shared application. If input events from multiple users are simultaneously applied to the shared application, event streams for non-atomic events (i.e., mouse drags) may become interleaved, and the result may not be what the users intended.

For example, in Fig1, A and B are sharing a drawing application, and they simultaneously attempt to draw separate freehand curves by dragging their mouse cursors. However, the location of the previous mouse-drag event for A is different from that of B. Unless some protection mechanism is enforced, A and B's mouse-drag events are interleaved and delivered to the application.

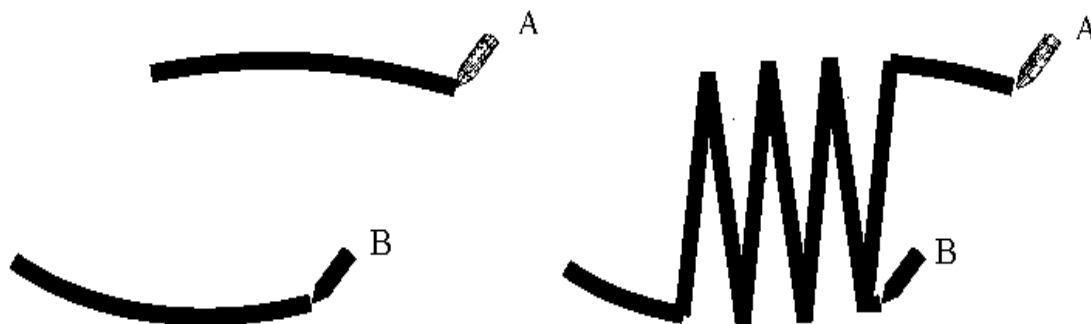


Figure 3. (left) shows what was intended, (right) shows the potential result of the interleaved mouse-drag events. The application has drawn a line between the alternating mouse positions of both A and B, instead between the sequential positions of B's mouse, followed by sequential position of A's mouse. This is an unexpected and undesirable result.

The floor control is used in collaboration transparency systems to avoid such conflicts, restricting input to only one participant at a time. Floor control is essentially locking at the coarsest possible granularity, the whole application.

In contrast to application wide locking, a collaboration aware application may also use locking, but at a finer granularity allowing collaborators to safely manipulate different portions of the shared data concurrently, such as paragraphs, sentences, or words in a text editor. Such fine-grained locking supports a more loosely coupled collaboration than floor control allows.

2.3.2 WYSIWIS feature

What You See Is What I See (WYSIWIS) is an abstraction that guides the design of multi user interfaces, it is an abstraction of some of the functional properties of a chalkboard in a meeting (cf. [29]).

Collaboration aware applications often support multiple modes of collaboration by relaxing WYSIWIS so that participant can simultaneously view different portions of shared data (cf. [16]), instead collaboration transparency systems impose severely strict WYSIWIS.

In "strict" WYSIWIS, everyone sees exactly the same image on their screen, and this also contributes to the imposition of tightly coupled collaboration because participants must view the same portion of the shared data simultaneously.

For example, in *Dialogo* all participants have identical layouts of the shared workspace and the strict WYSIWIS is guaranteed by delegating control over the shared application within the workspace. Also the *Microsoft NetMeeting* (cf. [24]) impose this degree of WYSIWIS and because of it, when a window or menu obstructs the shared application on the host machine, the corresponding portion on remote views is also obstructed. In the example showed in Fig4, the menu obstructs a segment of the application both in the central and the remote host. The menu is not displayed on the remote host because it is part of the application that is being shared. But this occlusion occurs only when it originates at the central host, for example, the pop-up menu, in the lower right in the second picture does not occlude the view of the application seen in the central host.

NOKIA Research Center
Giovanna Ferrari

April 2001

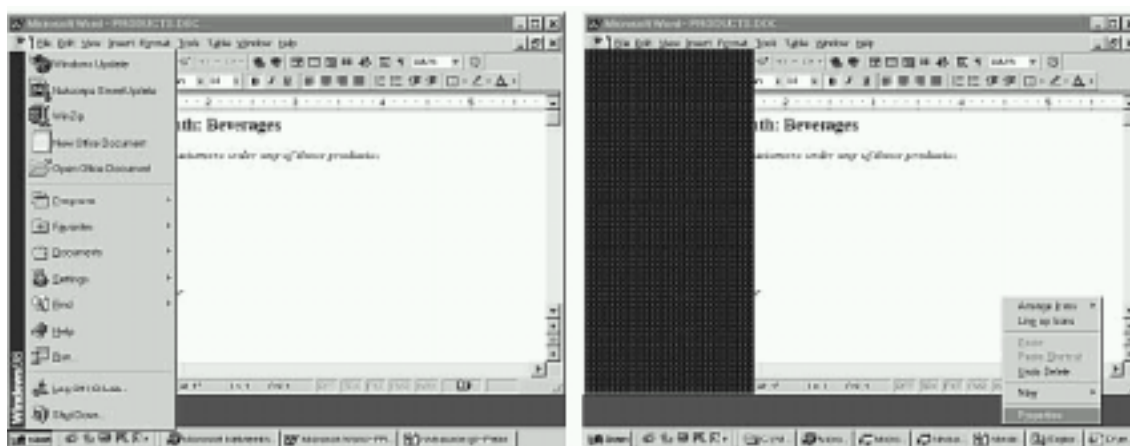


Figure 4. Two views of a shared application (Microsoft Word) in NetMeeting. The first is the screen on the central host, when a menu overlays the shared application, and the second is the screen on a remote host.

If the architecture used in collaboration transparency system is a centralized, display-broadcasting architecture (cf. [1] and [20]), participants receive only a graphical depiction of a central instance of the shared application. Therefore, participants see the same view of the application. Centralized display broadcasting is a principle cause of poor support of groupware principles, partially because it imposes strict WYSIWIS.

The difference in the degree of strict WYSIWIS is due to different means of obtaining the application graphics. In the severely strict case, the graphics data are obtained by directly inspecting pixels of the application host's screen buffer. Only graphics data associated with the shared application are distributed. For example the system only sends graphics data for windows of the shared application and not graphics from other windows, such as the obstructing menu. Another technique for display broadcasting is to intercept and distribute graphics as commands are sent from the application to the windowing system.

There are four dimensions, with corresponding constraints in the strict WYSIWIS definition: space, time, population, and congruence. Strict WYSIWIS applies to everything on the displays; applying it to only a subset of the visible objects (e.g., windows and cursors) relaxes the space constraint. Strict WYSIWIS requires that images are synchronized, allowing delays in updating or viewing relaxes the simultaneity constraint. Strict WYSIWIS requires shared viewing to apply to everyone in the full meeting group; allowing sharing to be limited to subgroups relaxes the population constraint. Strict WYSIWIS requires that images be identical; allowing alternative views (including visual variations) relaxes the congruence constraint.

2.3.3 Session

The timeframe within which a group of users meets in a distributed fashion is called a *session*. Here are defined the modalities of information exchange between users (cf. [7]).

One collaborative application can have many work sessions, and each session may have different users working in separated repositories, which means that some groups do not interfere with others.

Each session may contain such information as a list of participants, a description of them (e.g. location, picture, e-mail address), assigned role (e.g. coordinator, teacher, student), session resources (e.g. pointer, pen), floor control status.

All this information can be used to support collaboration awareness in the application or to assure synchronized access to the limited session resources.

Conference systems such as **Microsoft NetMeeting** allow several users to simultaneously share a group of applications. Users can enter work session or leave them. Other users can be invited to participate in the work session.

There are many approaches for the registration of the users. The easiest one is to made connections between participants during conference setup only, and all channels remain open until the meeting terminates. In the other side is the dynamic registration, where any participants should be able to join and leave the shared view conference at any time.

Sessions also handle the different roles of each attendee. Participants may, for example, have different levels of access to the meeting. These include no permission to join the conference, observer status, full read/write permission, and a special preferred status in floor control.

2.3.4 Gesturing

It's important allowing people to gesture to each other over the work surface. Gesture was observed to be a prominent communication channel in face-to-face work. Identifying who is active is important. Associating actions in the workspace with those initiating them plays a major role in using the workspace to mediate the group's interaction (e.g. turn-taking gestures) (cf. [31]).

Simple gesturing is possible when the shared view includes the controller's *cursor*. In most window systems, moving the cursor without depressing the mouse button does not generate any application input. For example a double click causes the cursor to flash, so the user can enter text in the cursor position, by the workstation keyboard.

Another more general approach is to provide a *telepointer*. It is a cursor displayed in each instance of a shared window and is used by cooperating users to refer to objects in the shared window (cf. [19]). A window might have one telepointer that is shared by all users, or each user could have their own telepointer that is somehow visually distinguishable from those of other users.

Most shared windows systems, for example **SharedX** (cf. [12]), include this kind of facility that displays an arrow-shaped pointer in the shared workspace. Each participant can turn their cursor into a telepointer on demand and gesture accordingly. More than one telepointer may be shown at a time, and the person using it does not have to be the application controller.

Equally valuable is the ability to make graphical annotation. Using a *tele-pencil* as opposed to a telepointer, a participant may write on top of the application. He may annotate portions of the view, circle items, or erase some or all of the marks.

3. MAJOR COMPONENTS

On the whole, any type of collaborative environment for remote applications consist of three major components (cf. [17]):

- It must have a synchronization mechanism that keeps all remote applications identical.

NOKIA Research Center
Giovanna Ferrari

April 2001

- It must have a cooperative/arbitration mechanism, or a protocol by which contention between remote collaborating applications is resolved.
- It must have a communication system that supports the two previous components and allows for a scalable and manageable cooperating session.

With respect to synchronization, there are three issues to consider. First, any action taken by one application must be reported to the others such that they may take the same action. This will be referred as MMS (Moment to Moment Synchronization) and is especially important for applications that maintain the WYSIWIS property, for example shared Whiteboard.

Second, an application joining an ongoing collaborative session must arrive at the same context as the other applications. This type of synchronization will be referred as SUS (Start Up Synchronization). Third, the MMS can begin only after the SUS sequence has finished.

With respect to arbitration mechanism, it applies the floor control policies, enforces mutual exclusion over shared objects, when they would not be modified by more than an application at a time, and is involved in avoidance or resolution of conflicts.

With respect to the communication system, a collaborative environment should support multicast communication, routing and flow control. Synchronization messages are of type many-to-many, which in turn justifies multicasting, this is also important for the transmission of multimedia activities. End-to-end routing and flow control must be carried out entirely by the communication system without the involvement of the applications. In turn, this lack of involvement by the collaborating applications is conducive to a scalable environment.

4. MOST KNOWN COLLABORATIVE SYSTEMS

A remote meeting tool is **RTCAL** (cf. [26]). It supports meeting scheduling by building a shared information space from participant's on-line calendars. During the meeting, each participant sees the shared calendar and their own private calendar. A chairperson oversees all activity, decides who has control of the shared view (each person can request control by entering a special control character), and is the only person who can terminate the conference. A small summary window indicates what the conference is about, who is present, who the chairperson is, and who currently has control. Dynamic registration of participants is allowed.

Another system is **Mblink** (cf. [27]) that allows multiple workstations to share a bitmap. It is unique in that all participants' cursors are visible on all the shared screens and distinctive cursors identify their owners.

A Macintosh-based suite of programs is **Cantata** (cf. [5]), featuring a sophisticated multi-person text-based "talk" system; a message broadcasting system and a shared view system. Additionally, a specialized tool called the *Participant Construct System* supports knowledge acquisition from groups of persons working cooperatively. The shared view subsystem, called *switchboard*, shares access to a participant's serial port through a terminal emulator, rather than sharing the standard applications running on the Macintosh.

Rather than share the complete screen, a person can selectively choose and share one or more "public" windows on the display through "window sharing". Hewlett-Packard's **SharedX** (cf. [12]) is one example of this genre. Based upon the X window system, it is a centralised window server that interposes itself between a running application and the

physical servers controlling each participant's actual workstation. It can be thought of as a switch that redirects X protocol to and from several X servers on behalf of applications. Registration is primitive. Through a "ShareTool", a participant can make a duplicate of her window appear on another person's display. Conversely, one may "pull" another person's window across to one's own display through a "PullTool". SharedX also provides a primitive protocol for manipulating floor control, on top of which several more sophisticated floor control interfaces have been constructed. Telepointers are also available. The idea of computer support for face-to-face meetings was pursued by Xerox PARC in an experimental meeting room for small groups known as **CoLab** (cf. [28]). The room is arranged with one workstation per participant, as well as a very large touch-sensitive screen and stand-up keyboard. CoLab employs customized, collaboration-aware software. Three systems were built: **Boardnoter**, a shared blackboard; **Cognoter**, a tool for brainstorming and idea organization; and **Argnoter**, a tool to organize and evaluate arguments. But in these systems, while a single large tele-pointer could be seen by all, individual cursors were not, neither did participants see each other's actions as they occurred, for actions were not broadcast until a complete graphical stroke was made or a complete text line entered (cf. [14]).

Another computer-supported meeting room, **Capture Lab** (cf. [8]) uses a simple view-sharing set-up. Each participant has a private workstation and has opportunity to gain serial control of a group of workstation running a large screen at the front of the room. Work from the private workstation may be copied and pasted to the group workstation and screen.

There are also many multimedia systems for sharing views. For example **Rapport** (cf. [10]) uses the X window system to support a multimedia conferencing system, or **EMCE** (cf. [11]), both integrate a voice channel along with the shared application. In the last one floor control is handled automatically by the system through a distributed dialogue-activated collision-sensing algorithm, which notes pauses and handles any contention for the floor. Other participants can also signal the current floor-holder when they would like a turn. **Rapport** also supports multiple telepointers, as well as pictures of the participants themselves. Participants can point to one another and "raise their hands" for attention.

Not all work is performed using a computer. One possibility allows participants to share video projections of a workspace, like with the prototype system of the Xerox PARC, called **VideoDraw** (cf. [32]) for video sharing of a virtual whiteboard between two participants. Each participant's work surface is a horizontally mounted video screen, covered by translucent drawing surface that can be marked with a pen. The video camera above the surface transmits the corresponding image to the other participant. The effect is a truly shared "whiteboard".

4.1 Summary table

In the table below are summarized the characteristics of all the mentioned systems:

	Collaboration Aware	Collaboration Transparent	Centralized Architecture	Replicated Architecture	Floor Control	WYSIWIS	Session	Telepointer	Pen	Cursor
GroupKit	X			X	X	R	X	X	X	X
NLS System		X	X			S		X		
Wscrawl	X		X		?	S	?	X	X	X
Dialogo		X		X	X	S	X	X	X	?
GroupSketch		X		X		R	X	X	X	X
GroupDraw		X		X		R	X	X	X	X
Microsoft Netmeeting		X	X		X	S	X	X	X	
SharedX		X	X		X	R	X	X	X	
RTCAL		X	X		X	S	X			X
Cantata		X	?	?	X	S	X	X		
Mblink		X	X		?	S	?	?	?	X
CoLab		X		X	X	S	X	X	X	X
CaptureLab		X	X		X	S	X	X	X	
EMCE		X		X	X	R	X	X	?	?
Rapport		X		X	X	R	X	X	X	
VideoDraw		X		X		R		X	X	X

5. REFERENCES

- [1] J. Begole, C.A Struble, C.A. Shaffer, "Leveraging Java applets: toward collaboration transparency in Java", *IEEE Internet Computing*, 1997
- [2] J. Begole, C. A. Struble, C. A. Shaffer, R. B. Smith, "Transparent sharing of Java applets: a replicated approach", *Proceedings of the 10th annual ACM symposium on User interface software and technology*, 1997
- [3] J. Begole, M. B. Rosson, C. A. Shaffer, "Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems", *ACM Trans. Comput.-Hum. Interact.*, June, 1999
- [4] J.Boyd, "Floor control policies in multi-user applications". *INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, 1993
- [5] E. Chang, R. Kaspersi, T. Copping, "Group coordination in participant systems". Technical report, Alberta Research Council, Alberta, Canada, September, 1987

NOKIA Research Center
Giovanna Ferrari

April 2001

- [6] P. Dewan, R. Choudhard, "Flexible user interface coupling in a collaborative system", *Human factors in computing systems conference proceedings on Reaching through technology*, 1991
- [7] H.P. Dommel, J.J. Garcia-Luna-Aceves, "A coordination architecture for Internet groupwork", *IEEE Proceedings of the 26 Th Euromicro Conference*, 2000
- [8] EDS "The Capture lab.", EDS Center for Machine Intelligence, Ann Arbour, Michigan, 1988. Videotape.
- [9] D. Engelbart, "Authorship provisions in AUGUMENT". In *Proceedings of the IEEE Compcon Conference*, 1984
- [10] J. Ensor, "Rapport: A multimedia conferencing system". *The ACM SIGGRAPH Video Review Supplement to Computer Graphics*, 45(5). ACM Press, Baltimore, MD, 1989. Videotape.
- [11] J.J. Garcia-Luna-Aceves, E.J. Craighill, R. Lang (1988), "An opens-system model for computer-supported collaboration". In *Proceeding of the IEEE Conference on Computer Workstations*, March, 1988
- [12] D. Garfinkel, P. Gust, M. Lemon, S. Lowder, "The SharedX multi-user interface user's guide, version 2.0" Technical report STL-TM-89-07, Hewlett-Packard Laboratories, Palo Alto, March, 1989
- [13] S. Greenberg, "Sharing views and interactions with single-user applications". In *Proceedings of the ACM/IEEE Conference on Office Information Systems*, Cambridge, Massachusetts, April, 1990
- [14] S. Greenberg, M. Roseman, D. Webster, D. Bohnet "Issues and Experiences Designing and Implementing Two Group Drawing Tools". In *Proceedings of the twenty-fifth Hawaii International Conference on System Sciences*, Hawaii, 1992
- [15] J. Grudin, "Computer-supported cooperative work: history and focus", *IEEE Computer*, 27 (5), May, 1994
- [16] C. Gutwin, M. Roseman, S. Greenberg, "A usability study of awareness widgets in a shared workspace groupware system", *Proceedings of the ACM 1996 conference on Computer supported cooperative work*, 1996
- [17] B. Ionescu, J. Binder, D. Ionescu, "A Distributed Architecture for Collaborative Applications", *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 1999
- [18] O. Jones, "Multidisplay Software in X: a Survey of Architectures", *The X Resource*, 1993
- [19] J.R. Kenneth, D.R. Olsen, "Smart telepointers: maintaining telepointer consistency in the presence of user interface customization", *ACM Trans. Graph.*, 13 (3), July, 1994
- [20] K. Lantz, "An experiment in integration multimedia conferencing". In *Proceedings of the conference on Computer-Supported Cooperative Work*, Austin, Texas, December, 1986
- [21] J.C. Lauwers, K.A. Lantz, "Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window system". In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Seattle, Washington, April, 1990
- [22] M.D.P. Leland, R.S. Fish, R.E. Kraunt, "Collaborative document production using Quilt". In *Proceedings of the ACM Conference for computer-Supported Cooperative Work*, Portland, Oregon, September, 1988

NOKIA Research Center
Giovanna Ferrari

April 2001

- [23] A. Lemke, N.A. Streitz, B. Wilson " WSCRAWL: The use of a shared workspace in a desktop conferencing System", GMD-IPSI, Tech. Report. Arbeitspapiere der GMD Nr. 672.
- [24] Microsoft NetMeeting, <http://www.microsoft.com/netmeeting>, Last accessed October,1998
- [25] M. Roseman, S.Greenberg, "GROUPLIT: a groupware toolkit for building real-time conferencing applications", *Conference proceedings on Computer-supported cooperative work*, 1992
- [26] S. Sarin, Greif I, "Computer-based real-time conferencing systems". *IEEE Computer*, 18(10), 1985
- [27] G. Singh, A. Gopalan, "A coordination service for distributed applications". In *Proceedings of the IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June, 1999
- [28] M. Stefik, G. Foster, D. Bobrow, K. Kahn, S. Lanning, L. Suchman, "Beyond the chalkboard: Computer support for collaboration and problem solving in meetings". *Communications of the ACM*, 30 (1), 1987
- [29] M.Stefik, D.G. Bobrow, G. Foster, S.Lanning, D. Tatar, "WYSIWIS Revised: Early experiences with Multi-user Interfaces", *ACM Transactions on Office Information Systems*, 5(2), 1987
- [30] A.S. Tanennbaum, "Operating Systems: Design and Implementation", *Prentice-Hall, New Jersey*, 1987
- [31] S. F. Ehrlich, T. Bikson, W. Mackay, J. C. Tang, "Tools for supporting cooperative work near and far: highlights from the CSCW conference", *Proceedings of the SIGCHI conference on Wings for the mind*, 1989
- [32] J.C. Tang, "Listening, drawing and gesturing in design: A study of the use of shared workspaces by design teams". *Xerox Technical Report SSL-89-3*. April, 1988