Multiuser Interface Design in CSCW Systems

Pedro Antunes, Nuno Guimaraes
Technical University of Lisboa - INESC *
e-mail:...paa@inesc.pt,nmg@inesc.pt

October 1994

Abstract

This document outlines some approaches, experiences and results of the user-interface design in several CSCW systems, highlighting a set of issues related with the successful user-interface design and defining a background where functional and innovative approaches to the development of synchronous cooperative systems will be experimented.

1 Introduction

The design and construction of cooperative applications poses a set of challenges at the user-interface design level. An appropriate design must consider the necessary mechanisms and schemes to offer an high degree of shared context, awareness and tailorability to cooperating users.

Following the common 2×2 matrix classification of CSCW [Johansen 91,Ellis 91, Applegate 91,Nunamaker 91], the user-interface design is especially relevant when synchronous cooperations and geographical dispersed interactions are considered. In this context strategies to limit the impact of the underlying communications system on the user interface functionality should be considered.

This document outlines some approaches, experiences and results of the user-interface design in several CSCW systems, defining a background where functional and innovative approaches to the development of synchronous cooperative systems will be experimented.

The outline of the document is the following. The issues relevant to the design of shared spaces are discussed in section 2, including the management of users'

*Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6º - 1000 Lisboa - Portugal, Tel: +351-1-3100000. Direct Line: 3100223, Fax: +351-1-525843. This work was partially supported by the CEC, through Esprit BR 6360 (Broadcast).

interactions, the display organization and the design constraints imposed by the architecture of the application. The combination of shared and private spaces is discussed on section 3. In section 4 the screen space management is considered. Section 5 deals with concurrency and concurrency control. Finally, in section 6, the design approaches to support awareness in cooperative work are presented.

2 Shared spaces

Shared spaces are the most noticeable way to support multiuser interactions in a shared environment based on personal workstations with graphical displays, keyboard and pointing devices.

Shared spaces are defined by [Sarin 85] as the parts of several users' graphical displays "in which everyone sees the same information" and by [Kamel 93] as "a shared, dynamically updated display with one or more control centers." This kind of functionality has been originally attributed to the NLS system dated from 1968 [Sarin 85,Greenberg 90,Engelbart 88b,Engelbart 88a].

The user-interface design issues posed by shared spaces are significantly different from the ones posed by a single user environment. In particular, the interaction modes must address multiple users' interactions, different paradigms for display organization must be supported, the management of the users' interactions must address the group perspective and several design constraints are imposed by the architecture of the cooperative application.

2.1 Interaction modes and WYSIWIS

WYSIWIS (What You See Is What I See) [Stefik 87] is a mode of interaction that assures that any of the shared spaces presents exactly the same information. This strict definition of WYSIWIS requires all attributes of shared spaces to be equal, not only the contents but also position, size, colors, scrollbar positions, iconification status, pointer position, etc. It is due to this strict definition that WYSIWIS gives a strong sense of shared context [Ellis 91] and allows users to be able to exercise contextual communication using terms like "this object" or "here."

Two issues should however be noted. First, experience with strict WYSIWIS has reported the need for looser coupling [Dewan 91], basically because cooperative work is a mix of public but also private work. Second, strict WYSIWIS requires a heavy use of the underlying communications system in order to propagate the users' inputs and outputs among the shared spaces. It has been reported that strict WYSIWIS is very difficult to implement efficiently [BL92].

WYSIWIS can be relaxed in several ways [Stefik 87,Ellis 91]. In the display space, by allowing each user to view different sets of objects in the shared spaces. In the time of display, by recognizing the problem posed by long delays in communications

or when reducing the communication load by compressing events. In subgroup population, by reducing the set of participants which require tightly coupling. And in congruence of view, by allowing personal customization of window attributes, like window position or size.

[NW92] defines WYSIWIMS (What You See Is What I May See) which allows independent viewing perspectives of the same canvas. WYSIWIMS is an extension of relaxed WYSIWIS in the display space, based on the use of viewports [Greenberg 90].

[Penz 93] defines WYGIWIG (What You Get Is What I Get) as a conceptual model that accepts momentary discrepancies in the different shared spaces but ensures that users are aware of the operations being carried out.

Many systems adopt one or several of these approaches. GROVE [Ellis 91] relaxes congruence of view by allowing different colors of objects in windows to reveal different read / write permissions. GroupDesign [BL92] has a time-relaxed WYSIWIS mode, which allows a user to suspend window modification notifications, and a display space relaxed WYSIWIS by allowing users to have their own view of the shared drawings. GroupDesign also uses a form of WYGIWIG by animating user operations like resizing in each space, instead of simply replicating the local users' inputs. [Cook 91] relaxes WYSIWIS in grain size by eliminating progressive changes, like the ones that occur when moving objects, and by not displaying on each shared space other cursors than the users' one. Suite [Dewan 91] allows flexible coupling of window properties in order to avoid "scroll wars" including the scrollbars, position, size and also menus.

2.2 Management of users' interactions

As we have seen previously, WYSIWIS, understood in its multiple forms of relaxation, is the fundamental paradigm used to describe the functionality of shared spaces in multiuser interfaces. Of course, the quality of the user-interface design for CSCW systems relies on the specific implementations of the WYSIWIS paradigm. WYSIWIS ensures some degree of shared context, awareness and tailorability but does not guarantee the appropriate management of the users' interactions. It is therefore very important to study the strategies adopted by several systems on the management of multiple users' interactions.

GROVE was built to allow everyone to see and edit everything without restrictions, relying on alternate communication lines (telephone) to overcome possible problems rather than restricting users' interactions. Experience with GROVE revealed that actions on the shared space are generally disruptive unless accompanied by verbal explanation but that collisions are surprisingly infrequent [Ellis 91].

RTCAL, on the contrary, allows input from one user only [Greenberg 90]. This approach requires implementing appropriate control mechanisms and policies which may be too restrictive compared to the subtle and natural control exerted in face-to-face meetings.

Suite [Dewan 91] allows users to define the coupling between shared spaces on a per object basis. Users can decide if changes to an object shall be transmitted to others, decide when to transmit and decide when to receive updates from other users. Objects are locked while being manipulated. This approach is similar to the previous one in the way that it only allows input from one user but restricts that functionality to several possibly locked objects rather than to one single shared space.

[Antunes 93,Antunes 91] addresses the separation between data and display objects, allowing multiple and possibly different views of the same application data. Users' interactions are mediated by the display objects and controlled by shared dialog objects. The dialog objects support different interaction styles and can be tailored by systems designers but not by end users.

[Bentley 92] describes "Cooperative User Display Agents" which are local to each user's workstation and manage the user's interactions with one object. The agents separate the application data from the user interface and encapsulate multiple display management of one data object. Users can interact directly with these agents for local tailorability of focus, visualization and position of each object.

Another issue associated with the management of users' interactions has to do with the pointers.

One approach, followed by MBlink [Sarin 85], is to implement multiple pointers on the shared space, which is done using low level graphical operations on each workstation's window.

Ensemble [NW92] supports multiple pointers but users can decide whether to export their pointer or not.

[Crowley 90] recommends, by experience, to restrict the number of pointers in shared spaces to two: the user's pointer and a shared telepointer. The telepointer eliminates the distraction found when many cursors are visible at once [NW92].

In MMConf [Crowley 90] the telepointer is activated by an user pressing the middle mouse button while moving the mouse. rIBIS [Rein 91] uses instead a group mouse which any user can grab and move using the workstation's mouse. The group mouse is automatically released if idle for more than five seconds.

TelePICTIVE [Miller 92] allows to activate the telepointer at the destination site, i.e. an user selects which user to monitor with the telepointer.

2.3 Display organization

Shared spaces do not have necessarily to conform to a flat surface where group work may be confusing, unfocused and chaotic [Ellis 91]. Several systems try to organize shared spaces in a way that keeps WYSIWIS properties but providing further display organization which allows individual users to create and follow parallel work patterns.

MObViews [Guimaraes 93] uses a matrix-like workspace with one active cell for each user. Only the active cells accept inputs from the associated users.

GROUPKIT [Roseman 92] uses the concept of "overlays," transparent windows that are placed above the main application window. Transparent windows have their own cursors and accept users' inputs in the form of graphical annotations placed on top of the main application graphics.

CaveDraw [Lu 91] has support for "transparent layers," which can be created on user demand and where all users can sketch. The transparent layers are stacked so that each previous layer is dimmed to a light color. Any user can filter the information displayed on the shared space by individually hiding or selecting transparent layers. [Lu 91] reports that this solution does not scale well since it is difficult to identify to which layer a drawing belongs.

ClearFace [Ishii 91] provides translucent, movable and resizable live face windows that can be placed over shared spaces.

LookingGlass [Scrivener 94] provides translucent live windows which are aligned with the working surface, allowing users to see each other as if looking through a window. Shared drawings are displayed above the translucent windows.

GroupDesign allows each user to select different modes of viewing the shared space. A localization mode synchronizes with another user and allows to monitor all the operations being carried out. An identification mode indicates with colors the other users' viewports over specific areas of the shared space.

[Kamel 93] present a classification of the display organization considering:

- Tiled
- User-configurable
- Overlayed – transparent, opaque, cyclic
- Intermingling – transparent, opaque, cyclic

2.4 Architectural issues

The design of shared spaces is not totally independent from the design of the cooperative application itself. In fact, any selected architecture imposes several design constraints which should be confronted with the intended functionalities of the user interface.

Three different architectures of cooperative applications have been found in the literature: multiple views of a single user application [Sarin 85,Greenberg 90], centralized multiuser application and distributed multiuser application [Lauwers 90a, Lauwers 90b] (some authors separate the distributed architecture in replicated and hybrid [Santos 93]).

One considerable design restriction posed by an architecture with multiple views of a single user application compared with the multiuser application involves the concept of collaboration-awareness. Single user applications are not aware of multiple users' interactions and therefore it is more difficult to customize the system [Bentley 92]. [Sarin 85] also reports that this kind of architecture cannot support multiple concurrent contexts, private views of shared information and that participants cannot be given different access privileges or environments. On such systems, the development has concentrated in the support for turn-taking protocols [Bentley 92]. Shared X [Greenberg 90] is an example of multiple views support to single user applications.

The discussion of centralized versus distributed architectures has for long been found in the literature [Lauwers 90a,Ahuja 90,Crowley 90,Antunes 91].

An advantage of the distributed approach is that it tends to generate relatively low network traffic [Ahuja 90] since outputs to users, which may be significant in graphics applications, are executed locally rather than being replicated to each site. Users' inputs can also be handled locally, thus reducing the network traffic. Of course, the advantage of the centralized approach is that the system does not need to maintain the consistency of the distributed data.

Tests with Rapport [Ahuja 90] revealed that the centralized version generated 3.6 times as many messages than the distributed version. For typical applications based on the X Window System this ratio raises to 6 times. Of course the distributed approach poses the problem of handling concurrency control which may require further communication. This issue will be discussed later.

The impact of the delays imposed by the communications infrastructure on the user interface is certainly one of the important issues in the design of shared spaces. That impact is reflected by the option for the distributed approach and on the different WYSIWIS modes of interaction which are a trade off between the performance of the communications infrastructure and the desired shared context and awareness.

Dialogo [Lauwers 90a,Greenberg 90], GroupDesign [BL92] and MMConf [Crowley 90] replicate the application at every site. Users' inputs are sent to all applications but the outputs are local.

[Bentley 92] describes a system where, instead of replicating one application in multiple sites, the application is distributed among multiple sites. In this system there is a separation between the user interface, which is distributed, and the application semantics which is centralized.

[Tatar 91] reports a problem in Cognoter about different delays in delivering text to users. The delays had to be balanced in order to avoid confusion of users having a particular text and others don't. This solution was pertinent in the specific supporting environment of Cognoter, a centralized application with the users face-to-face, but is not acceptable in the distributed approach since its main advantage resides in the more prompt individual feedback.

3 Shared and private spaces

Cooperative work implies that users have a shared context and be aware of other users' actions. Cooperative work does not require, however, that users be constantly under the same degree of coupling. In fact, the need for looser coupling has been reported [Dewan 91]. [Brothers 90] even reports that users appreciated as much freedom as the software could reasonably provide.

Reasons for loose coupling include: more efficiency in the handling of the tasks which can be decomposed in multiple parallel subtasks [Stefik 87]; the need for privacy during cooperative sessions, such as avoiding public embarrassment if the user is not familiar with the application [BL92,EK90]; allowing users to have different interests and viewpoints [Sarin 85]; and allowing easier social organization of the collaborative activities [Dourish 92].

Private spaces "present information that is only visible to the individual participants" [Sarin 85] and have been combined with shared spaces in several systems.

There are nevertheless several problems introduced by private spaces. Contextual communication is more complex, since users may be trying to reference to other users information that is only present in their private spaces. Several experiments with Cognoter [Tatar 91] revealed that users mistook references. The redesign of Cognoter, Cnoter, only allows users to move from public to private workspaces or vice-versa as a group rather than individually. This however requires strategies for discussion and agreement.

Also, "users felt a need to see things in the workspace that the system would not let them see" [Tatar 91]. Cognoter was designed for editing in private windows where only the finished text is broadcasted to participants. [Stefik 87] reports that users of Cognoter expressed frustration at not being able to see what others were doing. It is therefore important to users to know who is looking at what part of the shared space, to clearly distinguish private from shared spaces and to be able to align their views with other users' views.

The experience with TelePICTIVE has additionally shown the importance of carefully limiting the extent to which participants can branch into parallel subtasks due to difficulties in handling inconsistent changes in private spaces [Miller 92].

The usage of private spaces also makes it more difficult to save the shared environment [NW92].

In ShrEdit [Dourish 92] there is no distinction between shared and private spaces but there is an additional private window accompanying each shared window that reports information on other active users.

MObViews uses a spreadsheet approach, where one of the cells in the spreadsheet matrix can be selected and turned private.

In ABC [Jeffay 92] shared windows are managed within a virtual screen looking like another window.

[Cook 91] describes a system where shared and private spaces occupy the same physical space of the workstation but in fact are placed on two distinct layers, with shared views on the back and private views in front. Therefore, the underlying layer is kept consistent on all users' workstations. The system makes the distinction clear to users by using different visual cues.

Capture Lab [EK90] uses physical separation between private and shared spaces. Each user's workstation corresponds to a private workspace while a "public computer," with a large wall-mounted display, is the public workspace. Communication is done through a clipboard.

DOLPHIN [Haake 94] interconnects public liveboards with personal workstations. The shared space can be visible at the liveboards and at the personal workstations. Objects can be placed on the shared space either by direct hand-writing on the liveboard or by importing them from the personal workstations' private spaces. The system supports different types of objects, including sketches, terminal-typed text or structured documents.

4 Screen space management

Screen space management deals with tasks similar to the ones assigned to window managers, i.e. controlling the layout and size of user's windows. In an environment with shared spaces this may be a more complex problem since changes done by each user may have to be reflected on all users' screens according to the selected WYSIWIS modes of interaction. Window controls include window size, position on the screen, position relative to other windows, stacking order and iconification [Crowley 90].

Several relaxed WYSIWIS modes have been defined in order to ease individual screen space management. Cognoter allows different users to position shared windows at different places on the screen [Dewan 91]. In MMConf [Crowley 90, Lauwers 90b] position and stacking order are local decisions but window size and iconification are global to lessen the problems posed by contextual communication. Suite allows users to select the coupling of window size and position [Dewan 91].

When the environment contains a mixture of private and shared spaces it is necessary to offer some scheme for managing related windows cohesively [Greenberg 90]. The manager should distinguish shared from private windows, all windows associated with a particular conference and which of several conferences a window is associated with [Lauwers 90b].

Dialogo runs a window manager as a replicated application, ensuring that the same actions are taken on each user's workstation [Lauwers 90a]. In ABC [Jeffay 92] shared windows are managed within a virtual screen which is itself a window. Users may select a window manager for the virtual screen other than the one used for the rest of the screen.

Dialogo and Rapport use separate workspaces, "rooms," for organizing shared

and private windows [Lauwers 90b]. [Cook 91] describes a model with shared and private "perspectives" each one containing inside several shared or private views, respectively. Users can navigate and move views across perspectives through doors (an icon that looks like a door). Views gain the public/private characteristics of the hosting perspective.

5 Concurrency and concurrency control

The support of concurrency and concurrency control as requirements for allowing multiple users' interactions has a strong impact at the user level in terms of feedback and control [Rodden 91]. Several strategies have been devised in order to provide better user interfaces, awareness, and responsiveness of CSCW systems using the currently available technologies.

The first topic to be dealt with is parallelism, i.e. the support to multiple, simultaneous activities. We will also focus on the maintenance of data consistency in the presence of possible concurrent manipulations. Then, we will highlight the concurrency control mechanisms necessary to resolve the conflicts which result from parallel activities.

Two classes of approaches to concurrency control are considered, pessimistic and optimistic. The pessimistic approach to concurrency control tries to resolve conflicts prior to their occurrence while the optimistic approach is intended to resolve conflicts after such conflicts have been detected in the system.

5.1 Parallelism

If "one user's actions are not immediately seen by others, then the effect on the group's dynamics must be considered" [Ellis 91]. While in single-user applications the non-preemptive scheduling (where an action that follows an input proceeds until the control is released) is suitable for managing inputs, this may not be acceptable for multiuser inputs because lengthy computations cause delays on the responses to other users [Patterson 91].

RENDEZVOUS [Patterson 90] uses light-weight processes to implement preemptive scheduling and consequently allow interleaving of actions and prompt feedback to all users.

MMM [Bier 91] uses a different approach. Input notifications are broadcasted to a distributed application with timeouts for responses. If the timeout fires the destination process is marked as "sick" and further events to it are discarded until the process is marked as "healthy." The screen refresh on MMM requires that all data manipulations stop until a consistent view is produced on each user's display.

5.2 Data consistency

A requirement of cooperative systems is the preservation of the consistency of data in the presence of parallel and possibly conflicting users' manipulations.

If the application is centralized, like in Xtv [BL90], the input events are strictly serialized and handled by the centralized concurrency control mechanism. This solution, by requiring that each user input goes through the server, makes however the feedback slow. The system is also vulnerable to server crashes.

The distributed architecture corrects the slow feedback problem of the centralized model by caching data on each workstation [Stefik 87]. Feedback can then be produced locally. The distributed application is then required to serialize the users' input events and to keep the consistency of the distributed data.

In GroupDesign events are sent to replicas describing commands carried out by each user. The protocol is asynchronous because a replica never waits for a reply from other replicas. Logic clocks are used to ensure proper consistency.

The DistEdit [Knister 90] toolkit maintains a copy of the state of the editor for each user. The consistency is ensured by a reliable atomic multicast protocol.

Duplex [Pacull 94] uses a reliable atomic multicast protocol to maintain consistency and also allows discontinued operation in case of network partition.

In Ensemble [NW92] the application is distributed but the concurrency control is centralized. This ensures the data consistency. The advantage of having distribution is only on the local feedback of operations. Users see updates from other users as they are committed in order to maintain the shared context within reasonable bounds.

[Miller 92] argues that the level of locking should be controlled by users. Duplex gives to users the ability to specify the granularity of objects in order to reduce the probability of concurrent operations.

5.3 Pessimistic concurrency control

A pessimistic concurrency control mechanism avoids conflicts by requesting a lock before any data access.

Concurrency can be controlled with a floor-passing strategy which allows only one participant at a time to lock and manipulate the shared space[Sarin 85]. Both MMConf and rIBIS support this strategy but also provide less restrictive mechanisms for concurrency control.

In MMConf a floor manager centralizes floor requests and relinquishes. For fault tolerance, the manager may serve a floor request without receiving a relinquish.

With a centralized-lock strategy each computer has a copy of the data but cannot make changes to an item until it obtains ownership of that item [Stefik 87]. In Colab this approach has yielded unacceptable delays for obtaining locks.

Transaction mechanisms also support pessimistic concurrency control but have been reported as not well suited to interactive use due to limitations in response time and notification time [Ellis 91,Rodden 92].

[Barghouti 91] reports that "serializable concurrency control might decrease concurrency or, more significantly, actually prevent desirable forms of cooperation." CoVer [Haake 93] uses a versioning system for avoiding conflicts. The system monitors changes to objects and when detects conflicts it creates new versions of the objects and proceeds with parallel tasks.

[Barghouti 91] describes a "conversational transactions" model that adds version control primitives (checkout/checkin) to the versioning system. The system manages a public and several private databases allowing greater concurrency while ensuring the consistency.

5.4 Optimistic concurrency control

A full optimistic approach to concurrency control consists in allowing users to manipulate objects without any kind of locking, relying only on socially accepted practices to ensure consistency. In general, running open floor becomes increasingly difficult as the number of participants increases [Lauwers 90b]. In particular running open floor becomes increasingly difficult as the communication delay increases [Lauwers 90b].

With a "cooperative approach" [Stefik 87,Barghouti 91] data changes are broadcasted without any synchronization. If two participants make changes to the same data simultaneously there is a race. The idea behind this approach is to identify object manipulations on the shared space in a form that indicates to users the race conditions and rely on them to resolve the conflict. In GroupDesign if a locally conflicting operation was done previously to the global one, the local operation is discarded and the user notified.

The "dependency-detection approach" [Stefik 87] corrects the shortcomings of the cooperative one by detecting the conflicts and requesting human intervention. Useful information, like the author and time of changes is also provided to users.

[Sarin 85] and [Ellis 91] describe another mechanism where any workstation broadcasts changes but, after dependency detection signals a conflict, the changes are reversed. While workstations may temporarily hold inconsistent copies of data, such inconsistencies are quickly rectified.

Grove uses an "operations transformation" mechanism where changes are done locally and broadcasted for later dependency detection. If a conflict is found the operation is "transformed" in order to maintain the global consistency. The transformation depends on the type of operation (add, delete, etc.) and on the serialization of other concurrent operations [Ellis 91].

Duplex allows users to select between optimistic and pessimistic control on a per object basis. Since object granularity can also be chosen, the authors consider this

approach to be adaptive to users requirements.

[Barghouti 91] describes a "group paradigm" that hierarchically divides the problem of concurrency control in two: a group policy, considering small teams and possibly being optimistic, and a global policy that ensures the correct serialization of several groups and which may be pessimistic.

6 Awareness of cooperative work

In the course of everyday human interaction people use an extremely rich spectrum provided by intonation and/or body language and facial expressions to mediate feelings [Viller 91] and to synchronize the communication [Penz 93]. Gestures are specially relevant to focus attention and control the flow of a conversation [Greenberg 89].

This suggests the need for voice and video channels accompanying computer cooperations, as for instance in the TeamWorkStation [Ishii 91] or ARKola [Gaver 91]. If no such channels are available the system should at least offer each user some "awareness of others" as a form of compromise.

Furthermore, if computers mediate users' interactions there is also a legitimate request for providing to each cooperating user awareness of what other users are doing with the computer [Lee 90,Jirotka 91].

Another kind of awareness which computers can support is "group awareness," i.e. information directly delivered from a user to the group [Penz 93]. Telepointers are an example of support to this functionality.

6.1 Awareness of other users

The intention is to provide some "feeling of presence" to users as if they were in a face-to-face meeting.

Grove displays small faces of participants in a shared space. If users enter or leave the shared space their pictures appear and disappear accordingly.

FaceTalk [Penz 93] displays cartoon faces that users can configure to correspond to particular moods (agreement, disagreement, sadness, irritation, etc.).

ShrEdit has a control window which displays the names of participants in a session. Users can use this window to find or track other users on the shared space. [Dourish 92] reports that these features were not much used but considers that the problem was in the design of their user-interface.

In TelePICTIVE [Miller 92] each user is assigned a unique color which is displayed in any object that the user selects.

[Dourish 92] proposes using "change bars" indicating areas which have been

modified together with additional information like nature of changes and identity of who made the change. This information brings closer asynchronous work to synchronous work.

6.2 Awareness of other users' activities

[Nielsen 93] points out that users of cooperative settings are reluctant to expend effort in entering information on the computer for the purpose of being helpful to others. The awareness of other users' activities must be captured by the system.

Strict WYSIWIS is certainly the best way to provide awareness of others' activities. However, as [Ellis 91] points out, "a good group interface should depict overall group activity and at the same time not be overly distracting." Several systems have been able to implement more subtle variations of this kind of awareness.

In GroupDesign [BL92] an icon displayed over an object indicates that a change is about to be made by a user. While the icon is displayed the object is partially locked. GroupDesign also does a form of echoing one user's actions in other users' shared spaces. For instance, to move a rectangle, a "move" icon is first displayed on the other users' spaces and then, after the move is done, the icon is erased and the rectangle is smoothly moved to the new position. The advantage of this approach is that the move animations are done locally which simultaneously preserves awareness and reduces the communications load.

MULE [Pendergast 90] uses colors to identify locked text lines of a multiuser editor. The "free" lines are displayed in white while locked lines are displayed in yellow to the locking user and red to the other users.

GROVE uses a "cloudburst" model of text editing, where text changes color while being aged, which allows users to identify pieces of modified text.

GroupDesign also uses colors to display the age of objects, has a localization mode, which allows to visualize areas being edited by other users, has an identification mode which shows who created objects, and uses history to replay the last actions of the group.

6.3 Group awareness

Telepointers permit to point at information under discussion and develop meta-discussions [Crowley 90].

In Ensemble users can decide if they want to export the pointer or to import other pointers, which allows to customize individually the group awareness.

In MBlink a participant can see a pointer reporting the current mouse position and one other pointer that identifies the actual position as seen on the other users' workstations. The user can judge when the other participants' shared spaces will catch up and display the pointer at the same position [Sarin 85].

The graphical annotations over the main applications supported by GROUPKIT also qualify as group awareness.

VideoDraw mixes video with a drawing surface [Greenberg 90] which allows richer forms for delivering information to the group.

7 Conclusion

We have highlighted a set of issues related with the successful user-interface design of CSCW systems. From this description we present a summary table and observe the following issues:

- The functionality of shared spaces requires the selection of WYSIWIS modes of interaction, the management of the users' interactions, a display organization suitable to structure parallel work patterns, and careful study of the design constraints imposed by the architecture of the application.
- The combination of shared and private spaces requires adequate strategies to avoid degradation in shared context.
- The screen space manager should organize the individual environment, distinguishing shared from private spaces and providing tailorability.
- The system must support concurrency and concurrency control. Pessimistic and optimistic approaches can be considered.
- The system should enrich the communications medium with information commonly available in face-to-face interactions, including awareness of users, awareness of users' activities and group awareness.

Shared spaces	
Interaction modes	Strict WYSIWIS, relaxed WYSIWIS, WYSIWIMS, WYGIWIG.
Management of users interactions	No input restrictions, input from one user only, per object management, shared dialog objects, user display agents. Multiple pointers, user selected pointers, telepointers.
Display organization	Tiled, user-configurable, overlayed, intermingling, transparent, opaque, cyclic.
Architectural issues	Multiple views of single application, centralized multiuser application, distributed multiuser application.
Shared and private spaces	Accompanying windows, spreadsheet approach, virtual screen, distinct display layers, physical separation.
Screen space management	Replicated window manager, multiple window managers, rooms, perspectives.
Concurrency and concurrency control	
Parallelism	Light-weight processes, messages.
Data consistency	Centralized, logic clocks, reliable atomic multicast protocols.
Pessimistic concurrency control	Floor-passing, centralized-lock, transactions, versions, conversational transactions.
Optimistic concurrency control	Social practices, cooperative, dependency-detection, reversed changes, operations transformation, adaptive, group paradigm.
Awareness of cooperative work	
Awareness of others	Small faces, cartoon faces, control window, colors, change bars.
Awareness of other users' activities	Echoing of actions, locking colors, cloudburst model, aging colors.
Group awareness	Telepointers, graphical annotations, mixed video.

Summary table

References

[Ahuja 90] S. Ahuja, J. Ensor, and S. Lucco. A comparison of applications sharing mechanisms in real-time desktop conferencing systems. In *Proceedings of the Conference on Office Information Systems*, Boston, 1990.

[Antunes 91] P. Antunes, N. Guimaraes, and R. Nunes. Extending the user interface to the multiuser environment. In *ECSCW '91, CSCW Developers Workshop*, Amsterdam, September 1991.

[Antunes 93] P. Antunes and N. Guimaraes. A distributed model and architecture for interactive cooperation. In *Proceedings of the 4th Workshop*

on Future Trends of Distributed Computing Systems, Lisboa, Portugal, September 1993.

[Applegate 91] L. Applegate. Technology support for cooperative work: a framework for studying introduction and assimilation in organizations. *Journal of Organizational Computing*, 1(1), 1991.

[Barghouti 91] N. Barghouti and G. Kaiser. Concurrency control in advanced database systems. *ACM Computing Surveys*, 23(3):269–317, September 1991.

[Bentley 92] R. Bentley, T. Rodden, P. Sawyer, and I. Sommerville. An architecture for tailoring cooperative multi-user displays. In *Proceedings of ACM CSCW '92 Conference on Computer-Supported Cooperative Work*, Toronto, Canada, November 1992.

[Bier 91] E. Bier and S. Freeman. MMM: A user interface architecture for shared editors on a single screen. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM Press, 1991.

[BL90] M. Beaudouin-Lafon. Collaborative development of software. In *Proceedings of IFIP WG8.4 Conference on Multi-User Interfaces and Applications*, Crete, 1990. North-Holland.

[BL92] M. Beaudouin-Lafon and A. Karsenty. Transparency and awareness in a real-time groupware system. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Monterey, California, November 1992.

[Brothers 90] L. Brothers, V. Sembugamoorthy, and M. Muller. Icicle: Groupware for code inspection. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, Los Angeles, California, 1990. ACM Press.

[Cook 91] S. Cook, G. Birch, G. Birch, A. Murphy, and J. Woolsey. Modelling groupware in the electronic office. In S. Greenberg, editor, *Computer Supported Collaborative Work*. Academic Press, Inc., 1991.

[Crowley 90] T. Crowley, E. Baker, H. Forsdick, P. Milazzo, and R. Tomlinson. Mmconf: an infrastructure for building shared applications. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, Los Angeles, California, 1990. ACM Press.

[Dewan 91] P. Dewan. Flexible user interface coupling in collaborative systems. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 41–48, New Orleans, 1991. ACM Press.

[Dourish 92] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of ACM CSCW '92 Conference on Computer-Supported Cooperative Work*, pages 107–114, Toronto, Canada, November 1992.

[EK90] M. Elwart-Keys, D. Halonen, M. horton, R. Kass, and P. Scott. User interface requirements for face to face groupware. In *CHI '90: Conference on Human Factors in Computing Systems*, April 1990.

[Ellis 91] C. Ellis, S. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, 1991.

[Engelbart 88a] D. Engelbart. Authorship provisions in augment. In I. Greif, editor, *Computer-Supported Cooperative Work: a Book of Readings*, chapter 5. Morgan Kaufmann Publishers Inc, 1988.

[Engelbart 88b] D. Engelbart and W English. A research center for augmenting human intellect. In *Computer-Supported Cooperative Work: a Book of Readings*, chapter 4. Morgan Kaufmann Publishers Inc, 1988.

[Gaver 91] W. Gaver. Sound support for collaboration. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – ECSCW '91*, pages 293–308, 1991.

[Greenberg 89] S. Greenberg. The 1988 conference on computer-supported cooperative work: Trip report. *SIGCHI Bulletin*, 20(5):49–55, 1989.

[Greenberg 90] S. Greenberg. Sharing views and interactions with single-user applications. In *Proceedings of the Conference on Office Information Systems*, pages 227–237, Boston, 1990.

[Guimaraes 93] N. Guimaraes, P. Silva, J. Santos, and A. Siemaszko. MObViews: A multiuser worksheet for a mechanical engineering environment. In *2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE)*, Morgantown, WV, USA, April 1993.

[Haake 93] A. Haake and J. Haake. Take CoVer: Exploiting version support in cooperative systems. In *Human Factors in Computing Systems INTERCHI '93 Conference Proceedings*, Amsterdam, April 1993. Addison-Wesley.

[Haake 94] J. Haake, C. Neuwirth, and N. Streitz. Coexistence and transformation of informal and formal structures: Requirements for more flexible hypermedia systems. In *ACM European Conference on Hypermedia Technology ECHT '94*, Edinburg, Scotland, September 1994.

[Ishii 91] H. Ishii and K. Arita. ClearFace: Translucent multiuser interface for TeamWorkStation. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – ECSCW '91*, pages 163–174, 1991.

[Jeffay 92] K. Jeffay, J. Lin, J. Menges, F. Smith, and J. Smith. Architecture of the artifact-based collaboration system matrix. In *Proceedings of ACM CSCW '92 Conference on Computer-Supported Cooperative Work*, Toronto, Canada, November 1992.

[Jirotka 91] M. Jirotka, P. Luff, and N. Gilbert. Participation frameworks for computer mediated communication. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – EC-SCW '91*, Amsterdam, September 1991.

[Johansen 91] R. Johansen. Groupware: Future direction and wild cards. *Journal of Organizational Computing*, 2(1):219–227, 1991.

[Kamel 93] N. Kamel. An integrated approach to shared synchronous groupware workspaces. In *Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems*, Lisboa, Portugal, September 1993.

[Knister 90] M. Knister and A. Prakash. DistEdit: a distributed toolkit for supporting multiple group editors. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, Los Angeles, California, 1990. ACM Press.

[Lauwers 90a] J. Lauwers, T. Joseph, K. Lantz, and A. Romanow. Replicated architectures for shared window systems: a critique. In *Proceedings of the Conference on Office Information Systems*, Boston, 1990.

[Lauwers 90b] J. Lauwers and K. Lantz. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. In *CHI '90: Conference on Human Factors in Computing Systems*, Seattle, Washington, 1990. ACM Press.

[Lee 90] J. Lee. Xsketch: a multi-user sketching tool for X11. In *Proceedings of the Conference on Office Information Systems*, pages 169–173, Boston, 1990.

[Lu 91] I. Lu and M. Mantei. Idea management in a shared drawing tool. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – ECSCW '91*, pages 97–112, Amsterdam, 1991.

[Miller 92] D. Miller, J. Smith, and M. Muller. TelePICTIVE: Computer supported collaborative GUI, design for designers with diverse expertise. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Monterey, November 1992.

[Nielsen 93] J. Nielsen. Noncommand user interfaces. *Communications of the ACM*, 36(4):83–99, April 1993.

[Nunamaker 91] J. Nunamaker, A. Dennis, J. Valacich, D. Vogel, and J. George. Electronic meeting systems to support group work. *Communications of the ACM*, 34(7), July 1991.

[NW92] R. Newman-Wolfe, M. Webb, and M. Montes. Implicit locking in the Ensemble concurrent object-oriented graphics editor. In *Proceedings of ACM CSCW '92 Conference on Computer-Supported Cooperative Work*, pages 265–272, Toronto, Canada, November 1992.

[Pacull 94] F. Pacull, A. Sandoz, and A. Schiper. Duplex: A distributed collaborative editing environment in large scale. In *ACM 1994 Conference on Computer Supported Cooperative Work CSCW '94*, Chapel Hill, North Carolina, October 1994.

[Patterson 90] J. Patterson, R. Hill, S. Rohall, and W. Meeks. Rendezvous: an architecture for synchronous multi-user applications. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '90)*, Los Angeles, California, 1990. ACM Press.

[Patterson 91] J. Patterson. Comparing the programming demands of single-user and multi-user applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM Press, 1991.

[Pendergast 90] M. Pendergast. Design and implementation of a PC/LAN-based multi-user text editor. In *Proceedings of IFIP WG8.4 Conference on Multi-User Interfaces and Applications*, Crete, 1990. North-Holland.

[Penz 93] F. Penz, P. Antunes, and M. Fonseca. Feedback in computer supported cooperation systems: Example of the user interface design for a talk-like tool. In *12th Schaerding International Workshop, The Design of Computer Supported Cooperative Work and Groupware Systems*, Schaerding, Austria, June 1993. Elsevier Science.

[Rein 91] G. Rein and C. Ellis. rIBIS: A real-time group hypertext system. *Int. J. Man-Machine Studies*, 34(3):349–368, 1991.

[Rodden 91] T. Rodden and G. Blair. CSCW and distributed systems: the problem of control. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – ECSCW '91*, Amsterdam, 1991.

[Rodden 92] T. Rodden, J. Mariani, and G. Blair. Supporting cooperative applications. *Computer Supported Cooperative Work*, 1:41–67, 1992.

[Roseman 92] M. Roseman and S. Greenberg. GroupKit a groupware toolkit for building real-time conferencing applications. In *Proceedings of ACM CSCW '92 Conference on Computer-Supported Cooperative Work*, pages 43–50, Toronto, Canada, November 1992.

[Santos 93] A. Santos and A. Marcos. An algorithm and architecture to support cooperative multimedia editing. In *Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems*, Lisboa, Portugal, September 1993.

[Sarin 85] S. Sarin and I. Greif. Computer-based real-time conferencing systems. *IEEE Computer*, 18(10), October 1985.

[Scrivener 94] S. Scrivener, S. Clark, and N. Keen. The LookingGlass distributed shared workspace. *Computer Supported Cooperative Work*, 2:137–157, 1994.

[Stefik 87] M. Stefik, G. Foster, D. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1), 1987.

[Tatar 91] D. Tatar, G. Foster, and D. Bobrow. Design for conversation: Lessons from Cognoter. In S. Greenberg, editor, *Computer Supported Collaborative Work*, pages 55–79. Academic Press, Inc., 1991.

[Viller 91] S. Viller. The group facilitator: a CSCW perspective. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work – ECSCW '91*, pages 81–95, 1991.