User-Interface Support to Group Interaction

Pedro Antunes, Nuno Guimarães
Technical University of Lisboa - INESC [*]

April 1996

Abstract

This paper describes a user-interface system developed to support group interaction for same-time / different-place cooperative applications. The user-interface system is based on a model which defines four types of objects, each one dedicated to address a specific issue of group interaction support: information sharing, interaction control, structuring of group interactions and awareness of user activities in the system. Examples of combination of the above objects are given for the implementation of talk, joint editing and telepointer services.

1 Introduction

The computer support to group interaction consists of three basic functionalities: information sharing, control and user-interface. Information sharing allows to establish a common context between individuals. This functionality requires the specification of a data consistency model. Data consistency can be preserved through concurrency control mechanisms [7], e.g. locking, versioning, history, views, etc.

Group interaction adds the notion of interdependence [8] and coordination [13] to information sharing. Interdependence means that tasks flow from one individual to another while coordination introduces the requirement of interaction control. Several interaction control mechanisms have been proposed [17]: free mechanisms, that rely on the social protocols established by users and do not control the access to the medium, floor-control, semi-formal, based on language and formal mechanisms.

The user-interface is responsible for mediating users with the system. The user-interface must define a public space, shared by all users, maintain visual consistency of objects which are placed in the public space and, since group activities are assembled from a mixture of private and public activities, manage the interconnection of private and public spaces. One more user-interface requirement exists: it must provide feedback on information sharing, interaction control and management of public and private spaces.

The computer support to group interaction can also be characterised in time / space domains [16]. The combination of these domains defines four different types of systems: (1) same-time / same-place, which focus on the computer support to information sharing, since control and user-interface can be established face-to-face; (2) different-time / different-place, which minimises user-interface mechanisms, fundamentally because most work is done in the users' private spaces; (3) different-time / same-place, where few systems can be placed, minimises information sharing and interaction control, emphasising single-user interface aspects of

[*]Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6^o - 1000 Lisboa - Portugal, Tel: +351-1-3100000. Direct Line: +351-1-3100223, Fax: +351-1-3145843. E-mail: {paa,nmg}@inesc.pt.

Second International Workshop on Groupware, CRIWG '96. Puerto Varas, Chile, September, 1996.

interactions; and (4) same-time / different-place, which requires full group interaction support.

In same-time / different-place systems, information sharing is essential to preserve shared context between users that are not face-to-face; interaction control is essential to manage interventions by users that are simultaneously using the system; user-interface is essential to preserve the degree of co-presence of cooperative work. Facing this scenario, we focus in this paper on same-time / different-place support to group interaction.

This paper describes user-interface support to group interaction addressing the above issues. The system is based on a small set of graphical objects that ease the design of complex group interaction processes by decomposing and structuring information sharing and interaction control. Furthermore, the user-interface support delivers cues on users' activities and system operations, enhancing users' awareness of cooperative processes. The user-interface support was used to implement a same-time / different-place tool that runs group decision sessions structured according to several decision techniques developed in the social sciences field, such as Nominal Group Technique and Brainstorming [3, 2][1].

The paper is structured in the following way. We start describing the spatial model of the supported user-interfaces. Next section describes the graphical objects which mediate and structure users' interactions. Then, details on communication services required by the user-interface component are given. Examples of implementation of group interaction services using the user-interface support are also given. Finally, we give pointers to related works and present some conclusions.

[1] An implementation of the SISCO model [4] is also envisaged for the future.

2 Spatial model

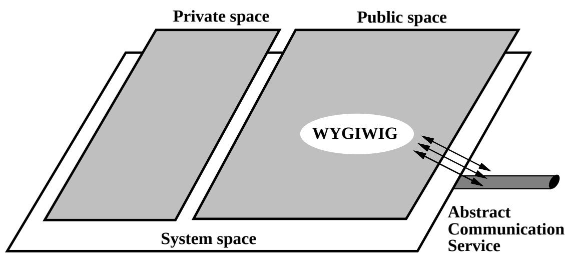

Figure 1: Spatial model

The spatial model describes properties of objects which appear in users' displays. The model is illustrated in Figure 1 and consists of private, public and system spaces. The private space is dedicated to support individual activities while the public space is dedicated to support group activities. The public space provides a common view to all users, implementing a WYGIWIG (What You Get Is What I Get) semantics [15]. This semantics ensures that, although many temporary inconsistencies arise in the public space (due to communication delays in the updates), the information that users get from the public space is common[2]. The system space is defined to provide continuity and allow transfer of objects between spaces. Objects may be moved across spaces (depending on the application semantics), changing their properties accordingly. The model provides one system space to each user with, at least, one private and one public space. Therefore, for n users there are n system spaces, n or more private spaces and 1 or more public spaces. Each space has the following properties:

- *System space.* Contains at least one private space and one public space. Allows to transfer objects between spaces and ensures that properties of moved objects change according to the spaces where they move to.
- *Private space.* Contains private objects, accessible to one single user.

[2] WYGIWIG relaxes the WYSIWIS (What You See Is What I See) semantics in the time domain.

- *Public space.* Contains public objects, accessible to all users according to the WYGIWIG semantics. This functionality is accomplished by replicating objects and broadcasting local changes to all replicas.

The communication facilities required by the properties of the public space are aggregated in what is called “Abstract Communication Service”. The Abstract Communication Service is described later.

3 Objects in spaces

There are four types of objects in spaces: Items, Assistants, Links and Awareness.

Items Items are repositories of “concrete” and durable information that one finds in group interactions: issues, positions, statements, etc. When placed in public spaces, Items share data between users. In private spaces, Items allow individual data creation and modification. An Item is assembled from *storage* (currently of text type), *representation* (icon and / or label) and *presentation* components. The *presentation* consists of a small window which displays the contents of *storage*. The *representation* is always visible on the space. To preserve display space, the *presentation* may not be visible. Users can apply the following direct actions on Items: open and close *presentation* (mouse “click” on the *representation*), edit *storage* (character input on *presentation*), move Item (mouse “drag”) and delete Item (menu driven).

Public Items acquire one more component: the *protocol*. The *protocol* specifies how replicated *storage*, *representation* and *presentation* are managed. Prior to any data modification, the Item checks the *protocol* state and executes any required operations. For instance, the locking *protocol* allows one single user – the locker – to execute actions on Item and denies any actions to other users. Figure 2 illustrates the locking *protocol* for the case where USER 1 has the lock which is requested by USER 2 to update the *storage*[3].

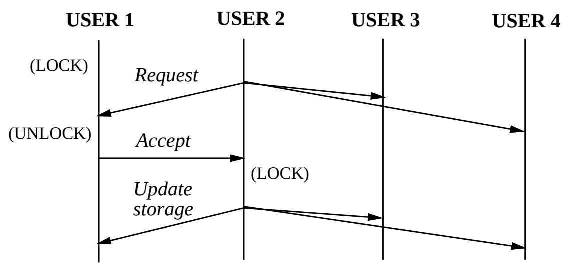

Figure 2: Locking protocol

Assistants Assistants are similar to Items but dedicated to represent and manage “intangible” and transient information found in group interactions: expression of ideas, comments, arguments, etc. Like Items, Assistants have *storage*, *representation* and *presentation* components and support the same user actions. Unlike Items, the *presentation* component of Assistants is intended to focus of the interaction process, i.e. it is the mechanism available in the system to communicate information and establish interdependence.

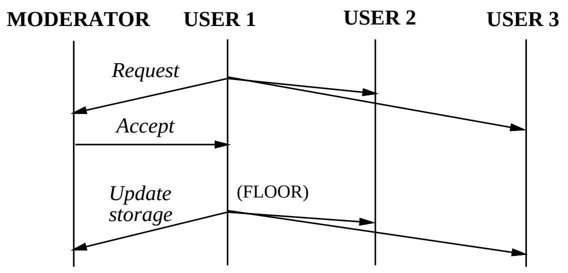

Figure 3: Conference protocol

Public Assistants also have *protocol* components. These *protocols* have a different nature from the Items’ *protocols*. They are dedicated to control interactions between users. For instance, the conference *protocol* manages the floor (access to the communication medium), granted by a moderator which circulates access between all users. The conference *protocol* is illustrated in Figure 3 for the case where USER 1 requests the floor.

[3] USER 1 unlocks the Item by closing its *presentation*.

Links	Meaning	Functionality
public Item → public Item *private Item → private Item*	Structure data	Connect Items
private Item → public Assistant	Expose private information	Copy *storage* to Assistant. Open Assistant's *presentation*
public Item → private Assistant	Acquire public information	Copy *storage* to Assistant. Open Assistant's *presentation*
private Assistant → public Item	Generate public information	Move *storage* to Item
public Assistant → private Item	Preserve public sentence	Copy *storage* to Item
private Assistant → private space	*private Assistant → private Item*	Create Item
public Assistant → private Assistant	*public Assistant → private Item → private Assistant*	Copy *storage* to Assistant. Open Assistant's *presentation*
private Assistant → public Assistant	*private Assistant → public Item → public Assistant*	Move *storage* to Assistant. Open Assistant's *presentation*

Table 1: Link actions

Links Links extend the number of actions supported by the user-interface system, allowing the association of two objects to one single action. Links can be established between two Items (*Item* ↔ *Item*), or one Assistant and one Item (*Assistant* ↔ *Item*). In the fist case, Links are established with the purpose of structuring data in spaces. In the second case, Links allow to structure users' interactions.

The Link *Assistant* ↔ *Item* operates in the following two steps: (1) acquires data from the origin object; (2) transfers data to the destination object. After these two steps, the Link has accomplished its task and vanishes. Note that all operations on public objects, including data acquisition and transfer, are ruled by their *protocol* components. Furthermore, since Items and Assistants can reside in private or public spaces, we can identify two interaction patterns supported by Links, which may emerge individually or coexist:

- *private* ↔ *public*
  Information exchange between users' private and public domains.

- *durable* ↔ *transient*
  Information exchange between the durable and transient contexts.

The several Links supported by the system are presented in Table 1.

Awareness Awareness objects provide graphical cues on users' activities and system operations. These elements are animated icons that appear and disappear according to the conditions they are associated to. Awareness is always associated to one Item or Assistant, appearing close to the element. The following awareness information is provided:

- *Consistency awareness.* Informs on the state of the *protocol* which manages replicated data (therefore this object is associated to Items only). Figure 4 illustrates this functionality for the locking *protocol*[4].

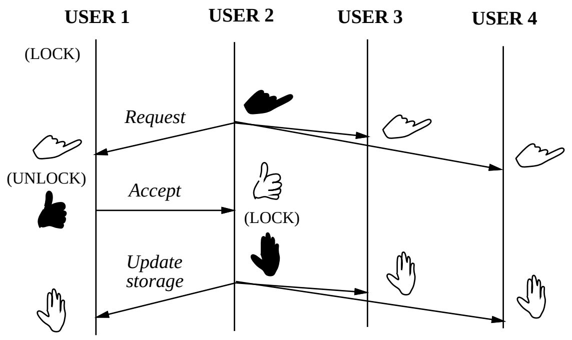

Figure 4: Consistency awareness

- *Interaction awareness.* Provides feedback on the state of the interaction control *pro-*

[4]Black icons represent users' own actions and blank icons represent other users' actions. The pointing hand represents a request, the thumb-up hand represents an acceptance and the open hand represents ownership.

tocol (associated with Assistants only).

- *Identity awareness.* Notifies the identity of the user that operated an Item or Assistant.
- *Time-elastic awareness.* Provides information on the functioning of the WYGIWIG semantics, notifying when all users effectively perceive the information delivered to the public space. This feedback is necessary for communications in large-scale networks, where users loose the notion of real-time, since messages may take an arbitrary time to be delivered [1]. Figure 5 illustrates this functionality.

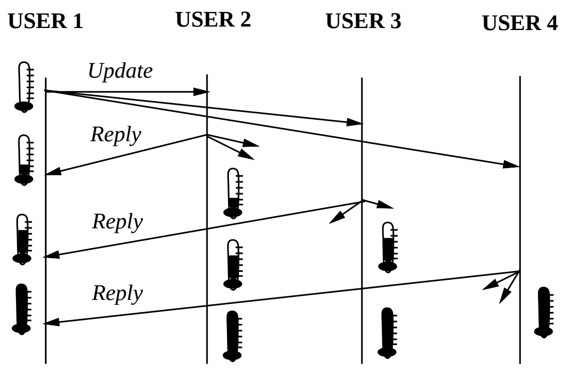

Figure 5: Time-elastic awareness

- *Disruption awareness.* Informs users when the WYGIWIG semantics cannot be maintained, i.e. broadcast messages do not reach all replicas, due to possible network partitions. Figure 6 illustrates this functionality.

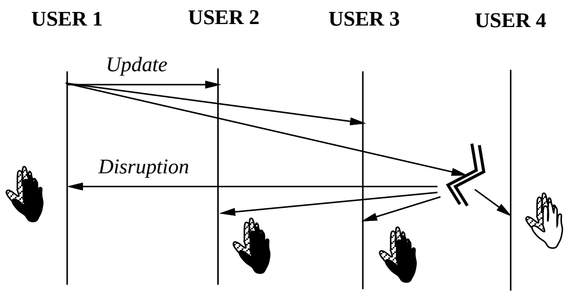

Figure 6: Disruption awareness

- *Application dependent.* Awareness objects can be programmed to provide application dependent information.

4 Abstract Communication Service

Public spaces require network communication in order to broadcast local object changes to all replicas. Furthermore, *protocols* also require communication between replicas. The Abstract Communication Service specifies messages that are communicated between replicas. The Abstract Communication Service does not make a distinction between Items, Assistants, Links and Awareness objects. It only recognises object names and positions. Messages exchanged through the Abstract Communication Service have the following format: `([addresses] [message])`.

The field `[addresses]` specifies the origin and destination of messages. Possible addresses are show in Table 2.

`user1.user2`	To `user2`
`user.ALL`	To all users
`user1.NOT.user2`	To all except `user2`
`SYSTEM.user`	From system

Table 2: Origin and destination of messages

Table 3 presents the messages defined by the Abstract Communication Service.

5 Examples

We present three common group interaction services experimented with the described user-interface system. The objective is to clarify how Items, Assistants, Links and Awareness are combined together to support group interaction.

5.1 Talk service

The basic objective of a talk service is to allow users to freely expose written statements.

`[Create|Move|Delete object [position]]`	Create, move or delete object
`Update object "text"`	Modify *storage*
`Open object`	Open *presentation*
`[Link|Unlink] object1 object2`	Link / unlink two objects
`[Request|Accept] object`	Used by the locking and conference *protocols*
`Done object user`	Required by time-elastic awareness
`Disruption [user*]`	Required by disruption awareness

Table 3: Messages defined by the Abstract Communication Service

This is accomplished with private and public Assistants, as illustrated in Figure 7.

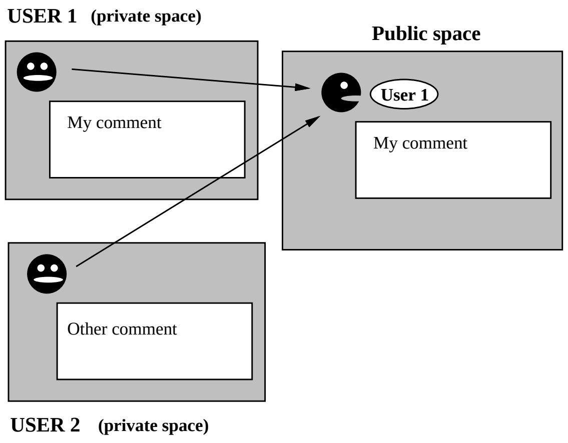

Figure 7: Talk service

- *Front-face*. Resides in the private space and allows users to write statements.

- *Side-face*. Resides in the public space and is the place where statements from users are displayed (through the *presentation*). To expose a comment, a user Links the *front-face* to the *side-face*.

An Awareness object is shown near the *side-face* with the identification of who exposed the statement. The *protocol* used by *side-face* is open-floor, which allows free access to the communication medium.

5.2 Joint editing service

The joint editing service allows several users to jointly create and structure a document. This service is illustrated in Figure 8. The joint editing service is based on dividing the document in parts (Items) and allowing individual work on each part.

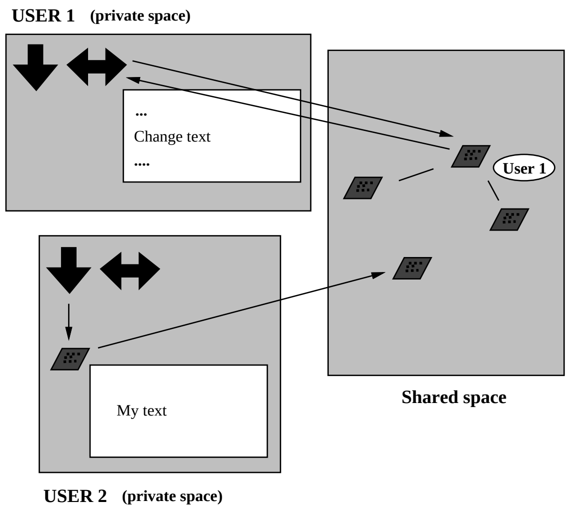

Figure 8: Joint editing service

Two types of Assistants coordinate user interventions in document parts:

- *Vertical-arrow*. Creates new document parts, when Linked to the private space (see example of USER 2 in Figure 8). New parts can be moved to the public space after being edited.

- *Horizontal-arrow*. Coordinates modifications in document parts. First, the user Links a document part to *horizontal-arrow*. The text is transferred to the Assistant and the *presentation* is opened in order to allow editing. Finally, the user Links the *horizontal-arrow* to the document part in public space (see example of USER 1 in Figure 8). This Link transfers the text, with modifications, back to

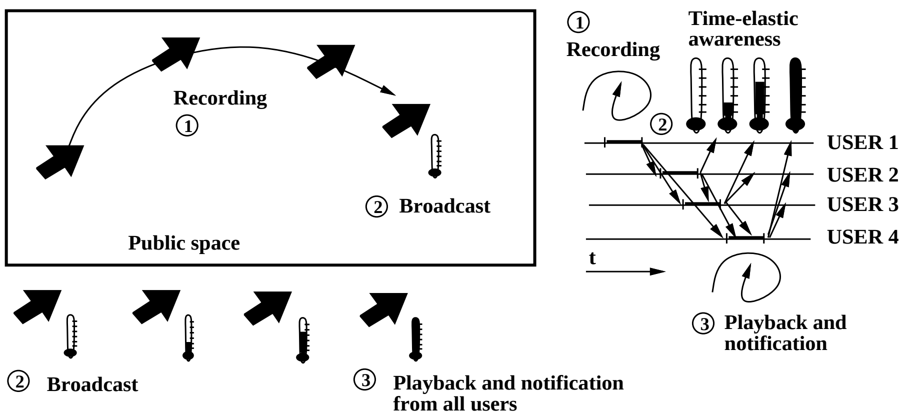

Figure 9: Shared context

the document.

Document parts use the locking *protocol* in order to avoid simultaneous modifications of their contents. This means that Links to a locked document part block while its contents is being modified. Awareness objects identify the users locking document parts.

5.3 Telepointer service

Telepointers are shared graphical entities designed to be manipulated by one participant at a time in order to point or get attention to a specific area or item of the public space [18]. The basic problem concerning the telepointer implementation is the lost shared context and consequent ambiguity in same-time / different-place settings, particularly with large-scale networks. To preserve the notion of movement, multiple messages have to be broadcasted to replicas. However, large scale networks are characterised by low throughput and long communication delays which turns the approach impracticable. We derive the telepointer from Item and implement the following functionality (see Figure 9).

1. The manipulation of the telepointer is governed by the locking *protocol*.
2. We avoid flooding the network with too many messages by recording telepointer movements. A recorded movement is broadcast (`Update` message) each time the locker stops moving the telepointer.
3. When each replica receives a recording, it executes a playback, i.e. replays the movement, and broadcasts a `Done` message notifying that the operation has been accomplished.
4. We use time-elastic awareness to inform users on the playback progress. Time-elastic awareness allows to preserve shared context independently of network delays.

After recording a gesture, the user must refrain from moving the telepointer until knowing that all participants play the gesture back. Only after the meter icon goes full, the user is sure that all participants have seen the telepointer movement.

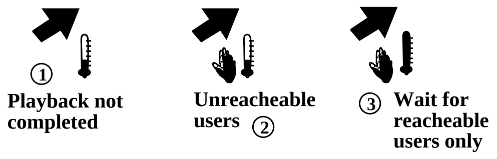

Figure 10: Loss of context

If one or more users become unreachable, due to a network partition, our strategy is to proceed with telepointer usage *but* inform all users about the loss of context.

The Abstract Communication Service notifies all telepointer's replicas, informing that is unable to communicate with some users (`Disruption` message). The notification identifies which users are unreachable. Then, each replica follows two possible strategies: (1) if a replica can reach the replica owning the lock, it proceeds with the locking *protocol* and preserves telepointer semantics; (2) if a replica cannot reach the replica owning the lock, it cancels the locking *protocol* in order to avoid blocking. A new locking *protocol* is started with the reachable users (if any) and a *temporary* designated locker. This approach guarantees that the telepointer semantics can be preserved but restricted to reachable users[5]. When users become reachable again, the telepointer eliminates the temporary lock and resumes the original locking *protocol*. users are aware of the different strategies and events by seeing different shadowed hands delivered by the Awareness object. This is shown in Figures 6 and 10.

6 Implementation

The user-interface system is implemented with the extension of a single-user graph editor developed in C++ and running on X Windows Systems / Unix workstations. Each workstation runs an instance of the editor. Shared and private spaces are implemented as rectangular areas of the graph editor. Items, Assistants, Links and Awareness are implemented with C++ classes derived from two basic classes: nodes and links. Two alternative communication systems were used to implement the Abstract Communication Service: (1) Mbus [11], a socket based communication service which provides message broadcast; and (2) NavTech / NavCoop [9], a group management platform which provides wide-area group communication protocols, replication management and network availability services.

[5]Unreachable users, as expected, are excluded from the time-elastic awareness mechanism.

7 Related work

A set of recent works can be related with some aspects of the user-interface system presented in this paper:

- *GroupDesk*. Is an environment for the coordination of cooperative document production that addresses the awareness requirement [10]. The solution is based on the distribution of object manipulation events and user registering of interests on particular events.

- *CoSARA*. Is a platform to specify and prototype multiuser interactions [20]. The focus is on data sharing (it implements dependency detection and locking protocols). The system also provides activity monitoring. The interaction mechanisms are defined at programming level, requiring knowledge of intricate specification models. CoSARA does not address coordination at user-interface level.

- *MEAD*. Is a system that supports the creation of multiuser interfaces [6]. The basic system component of MEAD is the User Display Agent which manages information display and multiple user interactions at the graphical object level. MEAD does not supply standard group interaction mechanisms, as the ones defined by Links.

- *DOLPHIN*. Is a groupware application that provides computer support to group meetings [19]. It defines private and shared spaces. The shared space supports whiteboard usage. DOLPHIN provides hypermedia objects and accepts pen-based inputs and gestures. Object transfers between public and private spaces use cut / paste operations. DOLPHIN relies on a centralised server for concurrency management, limited to the locking protocol. In remote cooperations, awareness is only supported through dedicated audio/video channels.

- *Contact*. Is a system to support cooperative writing [12]. Basically, this system coordinates user interventions in document parts. Editing is done with standard tools. The system creates Watch objects that inspect users' (Unix) commands on document parts. Users can infer other users' operations through Watch objects. The user-interface of Contact is based on a Web browser.

- *ESC*. Is a text-based environment that demonstrates the use of innovative communication channels for cooperation [14]. In some sense, the Assistants' functionalities are comparable to ESC's channel functionalities. For instance, possible communication channels include global, public, group and private clipboards.

- *DIVE/CyCo*. Are, respectively, three and two dimensional systems that support cooperative interactions [5]. These systems use a spatial model that defines levels of awareness based on spatial metrics (orientation and distance). Levels of awareness are used to fire different events. For instance, a far object is able to notice a shareable piece of text; becoming closer, the object is able to read text; even closer the object may be able to cooperatively manipulate text. Interaction is defined by moving objects.

- *COLA*. Is a system to support cooperative work [21]. COLA is based on a model which defines activities (cooperation), roles (interaction control) and events (awareness). The objectives presented in this paper are the same of COLA: support for cooperation, control and awareness. However, COLA does not address the user-interface level.

8 Conclusions

In this paper we present a system which supports the construction of user-interfaces for group interaction. The system addresses three common functionalities of this kind of applications: information sharing, interaction control and user-interface specific aspects. The proposed model is based on four object types: Items, which manage shared data; Assistants, which handle communication and coordination; Links, which are dedicated to structure complex group interactions; and Awareness objects, dedicated to deliver cues on users' activities and system operations. The system is well adapted to same-time/different-place interactions due to a careful consideration for the requirements of this kind of interactions. Some examples of group interaction mechanisms implemented with the system were presented, highlighting the flexible and structure abilities of the defined objects.

References

[1] P. Antunes, F. Cosquer, N. Guimaraes, and P. Verissimo. Adaptive group awareness for synchronous cooperation over large-scale networks. Submitted for publication., 1996.

[2] P. Antunes and N. Guimaraes. NGTool - exploring mechanisms of support to interactivity in the group process. In *First CYTED-RITOS International Workshop on Groupware CRIWG '95*, Lisboa, Portugal, September 1995. CYTED-RITOS.

[3] P. Antunes and N. Guimaraes. Structuring elements for group interaction. In *Second Conference on Concurrent Engineering, Research and Applications (CE95)*, Washington, DC, August 1995. Concurrent Technologies Corporation.

[4] G. Bellassai, M. Borges, D. Fuller, and J. Pino. SISCO: A tool to improve meetings productivity. In *First CYTED-RITOS International Workshop on Groupware CRIWG '95*, Lisboa, Portugal, September 1995. CYTED-RITOS.

[5] S. Benford. A spatial model of interaction in large virtual environments. In *Proceedings*

of the Third European Conference on Computer-Supported Cooperative Work – ECSCW '93, Milan, September 1993.

[6] R. Bentley, T. Rodden, P. Sawyer, and I. Sommerville. Architectural support for cooperative multiuser interfaces. *IEEE Computer*, May 1994.

[7] G. Blair and R. Rodden. The opportunities and challenges of CSCW. *Journal of Brazilian Computer Society*, 1(1), July 1994.

[8] R. Butler. *Designing Organizations*, chapter Requisite Decision-Making Capacity. Routledge, 1991.

[9] F. Cosquer, L. Rodrigues, and P. Verissimo. Using tailored failure suspectors to support distributed cooperative applications. In *Proceedings of the 7th International Conference on Parallel and Distributed Computing and Systems*, Washington, DC, October 1995.

[10] L. Fuchs, U. Pankoke-Babatz, and W. Prinz. Supporting cooperative awareness with local event mechanisms: The groupdesk system. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work – ECSCW '95*, Stockholm, Sweden, September 1995.

[11] S. Kaplan, A. Carrol, and K. MacGregor. Supporting collaborative processes with ConversationBuilder. In *Conference on Organizational Computing Systems*, pages 69–79, Atlanta, Georgia, November 1991.

[12] A. Kirby and T. Rodden. Contact: Support for distributed cooperative writing. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work – ECSCW '95*, Stockholm, Sweden, September 1995.

[13] T. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1), March 1994.

[14] D. Patel and S. Kalter. Low overhead, loosely coupled communication channels in collaboration. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work – ECSCW '93*, Milan, September 1993.

[15] 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.

[16] T. Rodden. A survey of cscw systems. *Interacting With Computers*, 3(3):319–353, 1991.

[17] 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.

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

[19] N. Streitz, J. Geissler, J. Haake, and J. Hol. DOLPHIN: Integrated meeting support across local and remote desktop environments and liveboards. In *ACM 1994 Conference on Computer Supported Cooperative Work CSCW '94*, Chapel Hill, North Carolina, October 1994.

[20] I. Tou, S. Berson, G. Estrin, Y. Eterovic, and E. Wu. Prototyping synchronous group applications. *IEEE Computer*, May 1994.

[21] J. Trevor, T. Rodden, and G. Blair. COLA: A lightweight platform for CSCW. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work – ECSCW '93*, Milan, September 1993.