A Distributed Model and Architecture for Interactive Cooperation

Pedro Antunes Nuno Guimaraes
IST/INESC
R. Alves Redol No. 6 1000 Lisboa, Portugal
e-mail: {paa,nmg}@inesc.pt

Abstract

Cooperation systems are intrinsically related with user interface systems. The design and implementation of a framework to support the construction of interactive and cooperative applications must be a consistent extension of the models and architectures of single user applications. The requirements for this extension include the management of the communication and coordination between users, the support for several levels of feedback (feedback about machine activities, individual user activities and group activities), the handling of the synchronisation issues.

This paper describes a distributed cooperation system based on a user interface toolkit and an interactive construction tool. The toolkit and tool were extended to enable communication, data sharing and coordination among several interacting users.

1 Introduction

Interactive systems and applications have evolved with the objective of providing better communication between human users and computing systems, by improving the usability of the user interfaces and increasing the bandwidth of the person-machine communication. Distributed systems have been evolving with the goal of increasing connectivity between computing systems, enlarging available resources and providing more flexible communication infrastructures. The availability of increasingly better user interfaces and powerful distribution platforms has certainly a strong impact on the development of Computer Supported Cooperative Work (CSCW).

Considering CSCW as any computer based system or tool designed to support users' cooperations leads us to a large set of approaches [9,8], merging technology, psychology and social sciences. Several systems are included in this broad category of tools, ranging from mail systems to conferencing facilities and multi-user editors. Several taxonomies have been proposed taking into account the simple time/space 2x2 matrix (*same time, same place, different time, different place*) [12], and other aspects of the tasks being collaboratively carried out [2,7].

Two other technological dimensions commonly used to classify CSCW systems are considered in this paper: *awareness* and *support*.

The awareness dimension classifies CSCW systems according to the degree of information the system is able to provide to users about the cooperative tasks. In the extremes of this taxonomy we find shared window systems [15], which integrate single user applications transparently, and electronic classroom systems [7] designed with explicit consideration for the collaborations that occur among users. In this later case, systems are classified as *collaboration aware* systems, as opposed to *collaboration transparent* [23]. Collaboration aware systems must support several levels of feedback, including feedback about machine activities, individual user activities and group activities [18].

The support dimension relates to the complexity of the technologies available for executing cooperative tasks; ranging from the simplest communication-only tasks, passing by information sharing tasks and ending in *process* tasks, which require communication, information sharing and also coordination and synchronisation [2].

Our objective is to design and implement a system which allows the easy prototyping of interactive and cooperative applications. This kind of applications requires both awareness and process support. The next section outlines the premises of our approach and underlying models. Section 3 presents the architecture that was designed to support the approach. The following section describes some examples that illustrate the mechanisms put into operation. The final sections open the discussion of some issues and enumerate some preliminary conclusions.

Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems, FTDCS '93. Lisboa, Portugal: IEEE CS Press, 1993, pp. 143-149. (ISBN: 0-8186-4430-3).

2 Rationale and models

The rationale behind our approach to the design of a *cooperation platform* can be summarised in the following principles:

> Interactive systems and applications for human-machine communication have been analyzed, designed and implemented according to models that structure the functionality and guide the architecture. Multiuser interfaces should not incur in fundamental conflicts with these models. Instead, they can be based on appropriate extensions of the previous models.

Single user interfaces define several levels of interaction, and group functions in broad categories usually defined as *Presentation*, *Dialog* and *Semantic Support*. An immediate extension of these categories to the multiuser environment leads us to define *Multiuser Presentation*, *Multiuser Dialog* and *Multiuser Semantic Support*.

> The current design and implementation of interactive applications reflects an integration of the User Interface (UI) models mentioned above, with object oriented methodologies and techniques. This approach contributes to the definition and design of encapsulated components that have clearly defined interfaces and functionalities.

Based on this notion, single user interactive applications are aggregates of objects, appropriately interconnected, that implement the desired interactions. Multiuser interfaces should also be designed as aggregates of objects, distributed among the contexts of the several users, interacting according to both single- and multiuser related rules.

However, it is not enough to distribute objects in order to allow *different place* interactions. These objects must also be distributed in the *different time* dimension.

Given these two broad principles, we will now briefly describe the single user interface model and the extensions which had to be introduced in order to support the different kinds of distribution.

User interface model

The stand-alone user interface environment is organised around an interactive construction tool that assembles interactive artifacts, i.e. groups of objects with distinct identity and presenting a given look, feel and functionality. The interactive artifacts are composed by aggregation of elementary objects, provided by a user interface toolkit.

Artifacts are assembled on-line and their functionality can be immediately parameterised and tested. This includes mechanisms to control user interactions, handle mid- to complex dialogues, execute some application-level computations (semantics) and interface with the rest of the application in a standard way. The artifacts can be stored for later retrieval, preserving the current state of the construction environment. The tool is also able to generate class-like persistent descriptions of artifacts for later reuse and multiple instantiation.

The objects provided by the toolkit are divided in four categories: *Display* objects encapsulate the input/output systems, managing the low-level interaction with the user (*Presentation*); *Dialog* objects manage the sequence of interaction elements; *Data* objects are abstract data types, providing common data manipulation operations and interfacing with the application-level operations (*Semantic support*).

Display and *Data* objects may be viewed as active variables which generate events when manipulated by the user or by the application, respectively. *Dialogs* and *Drivers* perform operations on active variables by receiving the events from *Display* and *Data* objects.

The global behaviour of an artifact is determined by the specific behaviour of the selected toolkit elements plus the links established between those elements. The links are event propagation channels between objects. See figure 1.

A complete description of both tool and toolkit can be found in [11].

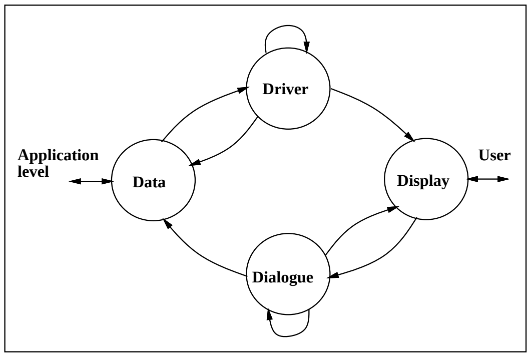

Figure 1: UI model

Multiuser interface model

As stated in the above principles, our approach to the design and implementation of a cooperation platform was to extend the single user models and mechanisms in the multiuser direction. According to the model presented above, there are clear extension paths:

- Distributed *Display* objects result in shared *Presentations*, in the line of shared window systems.
- Distributed *Dialogs* address the coordination of multiple user interactions.
- Distributed *Data* objects support information sharing of the application semantics and part of its state.

In this paper we did not selected the *Display* objects for multiuser extension. Not sharing *Display* objects results in the impossibility of implementing strict WYSIWIS multiuser interfaces. This has the advantage of avoiding concurrency control at the presentation level but imposes specific metaphors to designers and end users. Aspects related with these problems have been discussed in [10].

The multiuser model is similar to the previous single user model extended with artifacts that can be distributed through multiple sites by means of the distributed *Dialog* and *Data* objects. These extensions are implemented over a reliable multicast protocol, xAMp [21], and distribute objects by replicating their contents and broadcasting data changes.

Reviewing the CSCW dimensions described in the introduction, this multiuser model only addresses the space, support and awareness basic requirements. The higher-level CSCW functionalities, like coordination of users' activities, protocols for users' access to shared information or telepointers for joint discussions, are implemented at the toolkit level.

The time dimension is also not considered by the model. The functionalities related with this dimension are handled at the toolkit level by manipulating the consistency of the distributed objects. A strict consistency of the replicas gives the necessary support for *same time* operations, while relaxed (delayed or negotiated) consistency allows a quasi- *different time* approach. A full *different time* implementation is not currently implemented in the toolkit, mainly due to the lack of an appropriate mechanism and xAMp support. The next section delineates a different approach to this problem by allowing *different time* interactions between artifacts, instead of *different time* interactions within artifacts.

We stress that from the CSCW viewpoint the quality of the supported cooperative applications depends mostly on the higher-level functionalities implemented in the toolkit *Data* and *Dialog* objects. Some of these functionalities have not yet been identified but the currently implemented distributed objects rely on the following mechanisms:

- Replication – is the basic mechanism used to share application data and user interactions
- Locks – allow control of cooperation either by the application (implicit control) or by the user (control-free). At the lowest level locks are also used for synchronisation and concurrency control
- Relaxed consistency – gives autonomy to users' activities
- Strict consistency – for *same time* activities
- Pre-requisites – support coordinated users' activities
- Rules – support collaboration protocols

Some more implementation details relating integration of the toolkit with the xAMp can be found in [1].

3 Architecture

We found two reasons to provide further support in the cooperation platform. First, the UI construction tool should work well with the multiuser extensions, which is not a trivial task given that multiple tools and users are involved. Second, the necessity to create artifacts with high synchronisation and coordination within the aggregate is restricted to some CSCW tasks. In fact, cooperating users often require autonomy and tailorability [7,19]. This leads to the idea that artifacts are not frequently shared by the users but instead presented to users by other users and therefore synchronisation and coordination are done at a macro level.

Clearly some mechanisms to decentralise artifact construction, allow loose interactions between artifacts and the construction tool, and allow loose interactions between artifacts and application semantics are needed. The adopted system architecture takes this aspects in consideration by using *Multiuser Interface* (MUI) servers.

One server is attached to each user and mediates all her/his interactions through artifacts that the server is able to instantiate locally. The server has access to a communication medium to receive and send messages that control artifacts instantiation, deletion, queries about artifacts running in the server or handle artifact collisions (same objects with the same names).

The artifacts instantiation uses the mechanisms for storage/retrieval previously referred. The intended functionality is to allow users to create artifacts, using the UI construction tool, and send their descriptions within creation messages to remote servers. A successful interpretation of an artifact by a server results in its recreation on-site, providing the interaction with the user and the interface with the application level. This mechanism is similar to the one used by the NeWS window system [24], whereby *Postscript* programs can be loaded in the networked window servers.

We see the presence of an artifact in a MUI server as a request or solicitation for a user to perform a collaborative task. Presumably, several artifacts will be stacked on the user surface at a time. This will require appropriate management of the artifacts, for instance to allow browsing. To some extent, browsing and navigation across a group of artifacts requesting user actions is analogous to the navigation process that occurs in hypermedia systems [4].

The application level computations can be placed in one or several servers and share the communication medium used by the MUI servers. The interaction is done between artifacts and application servers and the MUI servers are transparent. This preserves the independence between artifacts and MUI servers.

The interface with the application level is guaranteed by *Data* objects that broadcast messages through the servers' communication medium to the interested application servers. This approach complements the replicated approach of the multiuser model being more oriented to the loose cooperations between users. Also, the communication between artifacts and application servers can be fully *different time*, e.g. one application server can be waiting for a message from an artifact that may have not been created.

The global view of the system shows users interacting with the construction tool and sending artifacts to other users, and users interacting with running artifacts and producing high level events which are delivered to remote application servers. See figure 2.

The current implementation of this system uses MBus from the ConversationBuilder [3,14] as the communication medium.

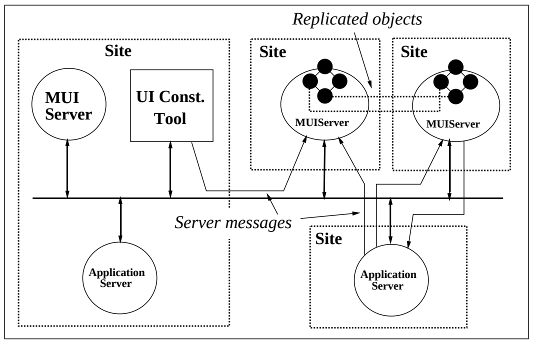

Figure 2: System architecture

4 Examples

Talk artifact

Figure 3 shows the aggregated objects that form a talk-like artifact. This artifact allows several users (two are shown in the figure) to enter text in areas visible to all but only writable by one of them. Replicated *Data* objects are needed in order to deliver the written text to the multiple *Display* objects.

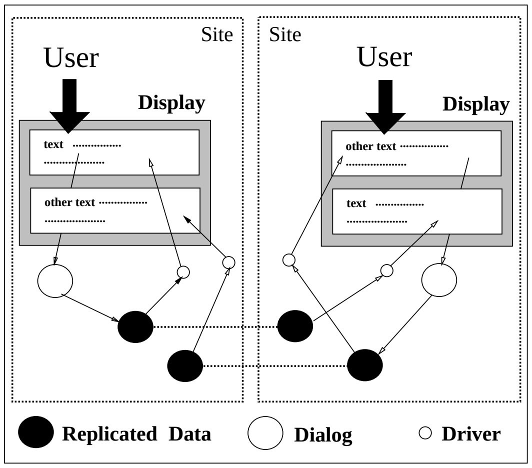

Figure 3: Talk artifact

Blackboard artifact

Figure 4 shows the internal structure of a blackboard artifact. Several users can write text on the same visible board of the artifact. The access to the board is controlled by one of the users, the one that has the floor control buttons. These buttons control the lock mechanism of the distributed *Dialog*. When the floor is taken only the locked replica accepts and delivers text entries. The *Data* and *Driver* objects are used as in the first example.

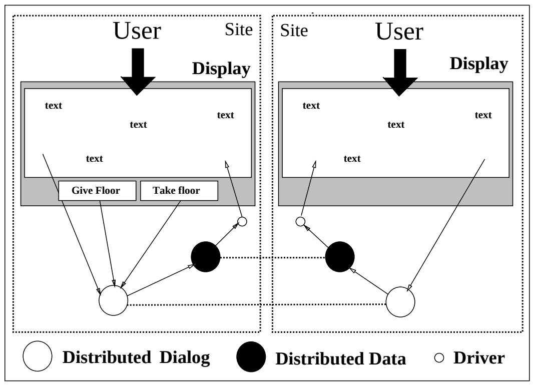

Figure 4: Blackboard artifact

Voting artifact

This example illustrates some of the combined UI construction tool and MUI Servers functionalities. The objective is to allow a user to propose one issue for voting.

The proponent uses the construction tool to assemble a vote form visually constituted by a Box with a Label and two Buttons. The figure 5 illustrates the internal structure and executed operations to design the artifact.

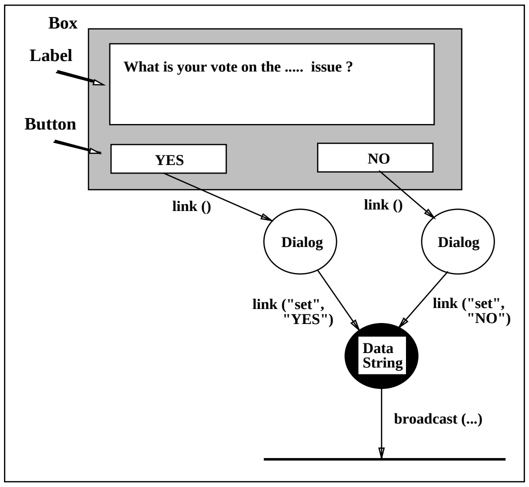

Figure 5: Voting artifact

First, the proponent creates the *Display* objects and places the issue on the Label and the tags "YES" and "NO" on the Buttons. A String *Data* object is instantiated and used to interface with the votes counter.

The "YES" ("NO") button is linked to a *Dialog* object and programmed to set the String contents to "YES" ("NO") whenever it receives an event. This simple mechanism sets the String to "YES" or "NO" depending on which button is pressed.

The String class selected from the toolkit broadcasts automatically the object contents to the MBus when it is updated. The broadcast message has the form: ("object-name" "YES") (we are not concerned here with the obvious security problems posed by this implementation).

After the above design operations, the proposer has to send the voting artifact to the users.

To collect the answers from the various vote artifacts the proposer must run a server that sits hearing the MBus and picking the specified messages. Currently this server is specified in C code using the MBus library. Further development will integrate a spreadsheet with MBus access that automatically collects the messages and creates cells with the message contents. The votes counting will then be a trivial spreadsheet operation.

5 Discussion and future work

The proposed technical approach has similarities with the Suite [5] and Rendezvous [17] systems, namely the mechanisms for updating replicated values and the decoupling of data and views. One aspect that should be stressed is the close relation between the toolkit and UI construction tool and therefore the new possibilities of its usage.

As [13] clearly states, one must provide powerful conceptual tools for integrating mechanisms and forms to support group work. This poses two challenges to the technical support for CSCW: the tools should be well integrated with the conceptual and social space of the collective and the tools should be well articulated with the mechanisms of group work.

The tool described in this paper is directly derived from the techniques for UI construction, which certainly has an important role in the cooperation process. However, other tools are certainly needed, like the Action Space, Node and Version browsers in ConversationBuilder [14].

Awareness is critical in CSCW systems [6]. The implemented low-lever mechanisms to support the various types of feedback are necessary but require further implementation at the toolkit level more oriented to users expectations and better integrated with the construction tools.

We believe that users familiarised with the current widget-type user interfaces can use artifacts as interactively generated, active and highly-structured conversation elements. The real usage of artifacts in real cooperative tasks must however be assessed. Also, the necessity to combine alternate channels for informal communication, like video or voice, must be studied.

The current system platform supports both prescriptive and non-prescriptive cooperative tasks. The rationale we followed while developing the toolkit was *to separate mechanisms from policies* [20] and therefore no policies are implemented at that level. However, because artifacts have some computational capabilities, it is possible to design (or re-design) them to prescribe specific interactions. As with the Oval *radically tailorable* tool [16], the end-user becomes also designer.

From the distributed systems support point of view, we found a lack of support to mixed *same time* and *different time* operations. This aspect and others related with a better integration between group technology and CSCW technology will be explored by the authors on the further development of the CSCW platform and also by the ROMANCE (Replicated Object MANagement Configurable Environment) project [22] at INESC.

6 Conclusion

We described a cooperation platform organised around a UI construction tool and toolkit. The single user interface models have been extended to include multiuser interactions.

The functionality of the system is based on artifacts that can be interactively created, parameterised and distributed to several users. Artifacts encapsulate multiuser look, feel and functionality and allow to combine several mechanisms for sharing data, control cooperation, maintain consistency and synchronise activities.

The usage of artifacts is decentralised in space and time. Users are allowed to disseminate artifacts through disperse sites and to interact with them on a discretionary basis.

Artifacts can be articulated and can interact with application servers allowing both structured and unstructured cooperations. Several examples of usage have been given, including a talk-like artifact, a blackboard artifact and a voting artifact.

References

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

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

[3] A. Carroll. *ConversationBuilder: Building Blocks for Open Collaborative Systems*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1992.

[4] J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, pages 17–41, September 1987.

[5] 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.

[6] 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.

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

[8] S. Greenberg. *Computer-supported Cooperative Work and Groupware*. Prentice-Hall, 1991.

[9] I. Greif. *Computer Supported Collaborative Work :A Book of Readings*. Morgan Kaufmann Publishers Inc, 1988.

[10] I. Greif and S. Sarin. Data sharing in group work. In *Computer-Supported Cooperative Work: a Book of Readings*, pages 477–508. Morgan Kaufmann Publishers Inc, 1988.

[11] N. Guimaraes, L. Carrico, and P. Antunes. Ingrid - an object oriented interface builder. In *Fifth International Conference on the Technology of Object-Oriented Languages and Systems TOOLS USA '91*, Santa Barbara, California, July 1991.

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

[13] P. Johnson-Lenz and T. Johnson-Lenz. Post-mechanistic groupware primitives: Rhythms, boundaries and containers. *Int. J. Man-Machine Studies*, 34(3):385–418, 1991.

[14] 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.

[15] J.C. 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*, pages 303–311, April 1990.

[16] T. Malone, K. Lai, and C. Fry. Experiments with Oval: A radically tailorable tool for cooperative work. In *Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work*, pages 289–297, Toronto, Canada, November 1992.

[17] J.F. Patterson, R.D. Hill, S.L. Rohall, and W.S. 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.

[18] 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, Design of Computer Supported Cooperative Work and Groupware Systems*, Schaerding, Austria, June 1993.

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

[20] 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.

[21] L. Rodrigues and P. Veríssimo. xAMp: a Multi-primitive Group Communications Service. In *Proceedings of the 11th Symposium on Reliable Distributed Systems*, Houston, Texas, October 1992.

[22] L. Rodrigues and P. Veríssimo. Replicated object management using group technology. In *Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems*, Lisboa, Portugal, September 1993.

[23] M. Stefik et al. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47, 1987.

[24] Sun Microsystems. *NeWS Technical Overview*, January 1988.