CYTED-RITOS International Workshop on Groupware, Lisbon, Portugal,
September 1995

A Group Based Approach to Build Flexible CSCW Infrastructures

Henrique João Domingos
hj@fct.unl.pt

José Legatheaux Martins
jalm@fct.unl.pt

New University of Lisbon
Dept. of Computer Science

Abstract

The recent multidisciplinary approach of CSCW shows a large variety of requirements and criteria associated to several applications. Flexibility is a common key-criterion and a fundamental issue associated to CSCW. To develop future collaborative applications, the programmer needs extensible underlying layers, highly reconfigurable, offering a high degree of flexibility.

The development of complex collaborative applications can benefit from group concepts: group communication, group management support and group oriented models. CSCW requires general purpose integration platforms structured on group abstractions to ease the development process. In this paper we consider the requirements of CSCW from the architectural support infrastructure viewpoint to propose an integration model based on the facilities offered by an extensively layered architecture providing an high degree of orthogonality and, consequently, high flexibility.

We discuss the use of group communication and its related protocols and concepts to build efficient general purpose platforms, considering the characteristics that such a group support should fulfil to make this approach realistic. Using this group based approach, we identify recent group development platforms as good candidates to support this research. Finally, we consider the need of programming abstractions to encapsulate group concepts.

Key words: *CSCW - Computer Supported Collaborative Work, Distributed Systems, Flexibility, Group Communication, Group Management, Replication, Group-Based Distributed Programming.*

1. Introduction

The worldwide nature of today's information infrastructures, has forced organizations to increasingly decentralise information and create new work methodologies, involving the active participation and cooperation of users and user's groups throughout computer networks. Those methodologies and their new interoperability concepts are emerging from the recent CSCW research and related experiences.

Recently, distributed applications are increasingly being used in new and innovative areas like: multimedia, networked large-scale information services, mobile computing and collaborative multi-user applications. CSCW, in complex distributed environments, involves many different requirements in a scenario of great integration and flexibility. In fact, CSCW takes a major role as

an innovative approach and an interdisciplinary research field. Cooperative work methodologies are introducing a large variety of new requirements associated with different aspects, including social and psychological approaches and new technological trends.

One of the most essential issue of CSCW is the subjacent group based communication concept. The idea of exploiting collaborative platforms based on group-oriented concepts and abstractions, emerges as an interesting approach to establish a new generation of system support including group communication protocols and management. The increasingly use of computers connected by large and progressive broadband networks infrastructures, reinforces those expectations in a context where scalability and fault-tolerance are now more than ever pre-requisites.

Group oriented programming concepts and models, providing management and coordination facilities, will be crucial to support strongly cooperative environments. Our approach stands by the use of group based support and tools, structured as middleware components, extensively layered, offering higher level communication abstractions and a high degree of flexibility and orthogonality.

We consider the requirements and taxonomy of groupware from that middleware support infrastructure point of view to describe the characteristics and properties of group communication layers to build CSCW platforms. Focusing on group communication protocols, group management tools, group semantics as well as its related concepts, our approach concludes that those platforms must be highly reconfigurable. We propose an integrated architectural model to easily support, in a flexible way, the groupware development, emphasizing the need of programming abstractions to encapsulate group concepts.

2. CSCW requirements

The identification of groupware application scenarios is very important to understand the amplitude of requirements. Different applications have different needs. Accordingly with the needs we can consider different categories of applications.

A generic support brought by different requirements imposed by the nature of different applications, it is a fundamental issue in CSCW. This interpretation stands for an architecture with high-level orthogonality, where underlying concepts don't restrict each other [Reinhard 94].

The principle of orthogonality is fundamental to the flexibility - a key-criterion, and it is crucial to the scalability - a key pre-requisite.

2.1 CSCW applications and the multidisciplinary trend

Different types of groupware applications have been developed. Some of them are research prototypes but we can also find many industrial products [Malm 94], including recent approaches on the characterization of multimedia supports [Williams 92] and distributed multi-party multimedia systems and applications [Szipersky 93].

[Cosquer 95] presents a useful survey on CSCW and related platforms covering not only the specific requirements and characteristics of groupware but also identifying the common factors involved in generic group oriented support infrastructures.

In different applications we can see significant variations in features, design and implementation strategies. Some examples are:

Electronic meetings: group participants discussing issues or making decisions based on certain coordination rules and voting processes.

Decision-oriented support: multi group decision support systems (MGDSS) provide facilities for group decision making where people, working together, choose collaboratively among different alternatives. These systems can be close to electronic meetings but involve problem-

solving work. Problem solving comprises different general-purpose tasks like: idea generation, submission of proposals, evaluation, negotiation, etc.

Group scheduling: management of group activities possibly based on group member's individual calendars or agendas. Shared calendars and agendas give users the possibility to manage in a cooperative way different events like: meetings, appointments, birthdays, etc.

Cooperative editing: cooperative document edition and composition involving shared texts, spreadsheets, drawings, etc.

Conferencing: conferencing facilities (involving from text data to multimedia data).

Workflow management: managing electronic documents and dataflows among different agents, possibly structured with some hierarchical criteria in the context of organizations.

Electronic Mail: extension of the common electronic mail service with group management. Involves group oriented functions to manage participants, reception notification, security and privacy guaranties, message registration, mailing lists, etc.

Concurrent Engineering (Computer Support for Concurrent Engineering CSCE): designates the activity of a possibly geographical distributed team of people, aiming at the concurrent design of a common project [Cosquer 95]. Involves structuring of activities, synchronization points between different phases, negotiation of activities and interchange of ideas in order to help in the development of better "quality versus cost" relations.

Shared window systems: multi-user sharing of window oriented workspaces.

Interactive Group Games: comprises similar aspects compared with the above. The competition involved and special diversity of operations and interaction with the environment can involve, however, some particular trends.

Examples above are representative and show that CSCW must be considered to support several multi-user cooperation [Greif 88],[Greenberg 91]. This results in a space of emergent interaction bringing together researchers and developers who share some particular interests (from human and social sciences to recent technological disciplines) with the inherent difficulties associated [Gruddin 94].

2.2 Taxonomy, requirements and criteria

Many authors have proposed a CSCW taxonomies to define, describe and evaluate the system requirements and tools, their relationship with collaborative processes and specific design alternatives [Reinhard 94], [Volksen 93], [Jarcsyk 92]. [Cosquer 95] proposes a classification of groupware platform, based on a combination of common and general characteristics, the type of underlying support and the role of group oriented support.

Requirements of CSCW can be sub-divided accordingly to different criteria at different levels. We will consider the following three levels: application, functional requirements and the system support.

2.2.1 Application level

In this level we consider the nature of each particular application seen in its usage perspective. There are two distinct orientations: (1) provide group extensions to single-user applications or (2) implement groupware in a full group environment of sharable objects.

The first approach provides essentially multiuser access to single-user applications. The idea is to establish group contexts adding collaborative features to well-known single-user applications. The second approach has a larger objective. Cooperation is established by means of a virtual common workspace, accessible through several sharable tools from a group-oriented interface.

Independently of the orientation, we must consider an application domain that can be characterized by three fundamental dimensions contributing to the complexity of the system: space, time and functionality.

Space is related with the spatial distribution involving the number of active participants and the organizational environment. Time is related with the period of time in which all the application events occur. Functionality is a measure of the global functional requirements, comprising the interaction level of cooperating activities, the *awareness* [Stefik 87] involved and the degree of coordination control. Functionality is more essentially related with the set of all the functional requirements associated with different cooperating scenarios.

2.2.2 Functional requirements

The following functional requirements are involved (with different complexity) in CSCW applications: participation, interaction, coordination, distribution, event notification, visualization, and privacy.

Participation is concerned with mechanisms used to define the way how a participants interacts within the application. Participation involves functions like: how a user join or withdraw freely, how to establish a limited number of participants, the way to force out a participant, definition of time constraints or other defined resource constraints, etc.

Interaction deals with communications and interactions between entities. The following requirements directly influence the communication models and interaction properties:

Interaction mode	• synchronous • asynchronous
Interaction content	• implicit • explicit
Interaction nature	• formal • informal
Information dissemination	• individual to individual • individual to group and vice-versa • individual to all and vice-versa • group to group • group to all

Table I - Communication models and interaction properties

The interaction nature depends on the required response time and the session control needs. The mode of interaction depends on the shared objects and their characteristics (ex: text or images). The interaction form can be associated with the establishment of some basic rules of acquaintanceship (ex: formal procedures in an hierarchical group of people or informal mechanisms as in a joint session of similar partners more oriented to peer-to-peer relationships).

The information dissemination is directly concerned with the group communication support.

Coordination is related with the mechanisms and rules created to support resource sharing. The complexity of concurrency control is intrinsic to the coordination nature of the application. It depends on the number of participants, group's size and the way individual members can interact or prefer to interact. Complexity of concurrency control depends on the granularity level of the objects involved: bytes, data-structures, file chunks, files, directories, libraries, processes, users, machines or devices.

Distribution is concerned with a strategy to design and implement applications (or some of their components). In the same application some functions can be centralized while others can

be distributed. A distributed strategy can be based in object replication with different consistency criteria. In general, a distributed strategy is more difficult to implement but has more potential advantages in terms of scale, performance and fault-tolerance. The use of replication avoids the existence of central points of failures. Replication strategies can be used to support fault tolerance, dynamic reconfigurations and automatic recovering. These properties are associated to the robustness characteristic of each application.

Event notification depends on the awareness and is related with system and application feedback. One of the approaches to build a CSCW system is to distribute single-user applications via some kind of sharing support, e.g. a distributed file system or a distributed shared memory layer. In such a system, each interaction is handled the same way regardless of individual user's roles. The users work with a familiar user interface and access existing, unmodified application programs. Some authors have classified this kind of groupware design as *collaboration transparent applications*.. [Stefik 87], [Reinhard 94].

However, in certain circumstances, users desire specific system reactions and notifications from the application (periodically or permanently), depending on their individual roles or rights. This is called *awareness* and systems with this functionality are usually called *collaboration aware systems*. [Stefik 87]. In this case, several levels of feedback must be available to all participants [Reinhard 94], [Penz 93]. Awareness seems to be crucial in the design of cooperative multiuser interfaces. Event notification is relevant in this case to establish criteria of user-reactive control. Mechanisms providing the necessary support for rapid prototyping design and implementation of user-reactive control have been referred as an important issue by many authors [Antunes 95], [Bentley 94]. In fact, many interactions in collaborative environments are event driven and some computations are close to reactive response models.

Vizualization is concerned with presentation criteria. In the case of collaboration-transparent CSCW systems and applications, the pure "what-you-see-is-what-I-see" paradigm determines a similar vizualization of shared data. Usually, users are interested in some variations concerned with the adoption of individual layouts showing identical contents of information. Is frequent the need of different objects of shared data or even different data granularity. A separation of the vizualization functions from the cooperating shared data is an important issue. External presentation levels and a multiple-fine grain coordination control must be also provided.

Privacy is related with security. The separation of private from public data in groupware applications is an important feature of collaborative environments. Private data should be classified as such and should only be accessible to certain users or groups, for example using access control lists or capability based policies managed in a group perspective.

We call *functional flexibility* to the global contribution of the above requirements.

2.2.3 Support level

Support level is the technological basis to support the functional requirements. There are two different initial approaches to build a generic support level. One way is based on extensions provided by multiuser windowing systems in order to give a generic multiuser collaboration support. This graphical approach to build multiuser collaborative interfaces is present in several existent applications. The other way, is to provide a generic system level group support for cooperation. This approach is based on the emerging results of distributed systems research.

The development of CSCW applications based on an hybrid approach is now usual. In fact the complexity of cooperation and shared objects (resources and data) manipulated in the context of multiuser interfaces needs new system level facilities. However, the nature of cooperation environments and the characteristics of human interaction factors also acts on the way how to structure the distributed systems design.

From our point of view, we are particularly interested in the "distributed systems" approach considering that a generic group support can be provided at system level or at user level by means of

a specialized group support layer. In this design, the separation of user-interface and application specific components is fundamental.

The "distributed systems" approach has advantages given the following scenarios:

- The new trends of computer utilization shows that scalability and mobility is an inevitable direction;
- The re-utilization of many existent applications in future cooperation environments is highly desirable;
- Aspects like: reusability, portability, heterogeneity support, customization and tuning must be addressed.

We can subdivide the support level in two main parts: the *system level* and the *CSCW middleware level*. The former is central to directly support the functional requirements.

The system level comprises well known components: I/O devices, network, basic communication protocols, window systems and graphical user interfaces. The communication layer subjacent to the system level involves LAN, MAN and WAN infrastructures and their protocols. To cover the indefinite area in which different participants of a groupware application are distributed, the support model must provide adaptive communication protocols to different transmission media.

According to [Reinhard 94], CSCW applications can be structured in four main components: input, output, application and cooperative data. The middleware level must support, in a flexible way, the implementation and interoperability of those components. Depending on the functional flexibility, each one of the four application components has specific requirements. One particular criterion can be present in one component while omitted in another. For example, each component can be replicated or centralized. Centralizing the data, guarantees more consistency, despite high traffic to the application or I/O devices. A replication policy requires less traffic for updates and provides a higher degree of fault tolerance. However, consistency is more difficult to achieve. The implications must be analyzed accordingly with each application component.

2.3 Flexibility

Flexibility must be considered as a common pre-requisite and a key-criterion orthogonal to all above requirements. The implementation of generic platforms providing the necessary support to all the requirements identified, must be evaluated based on this key criterion.

If we consider a taxonomy as a multidimensional space of requirements where each criterion corresponds to one dimension, a given set of CSCW properties corresponds to a flexibility vector in the application domain of time vs. space vs. functionality.

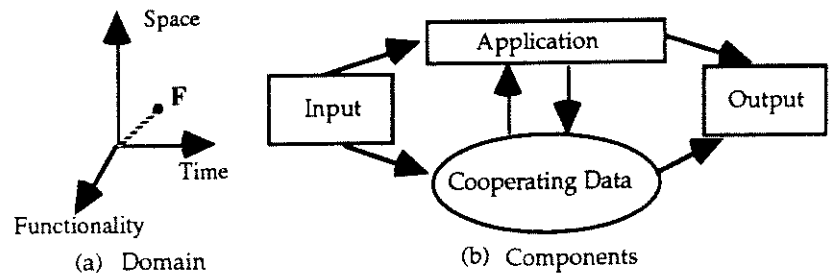

Fig.1 Application level criteria: (a) dimensions involved; (b) fundamental components

To implement a CSCW support level, the possibility to customize any given flexibility vector F (figure 1) must be considered. To provide this adaptability, orthogonal properties must be stressed, e.g. the supports must be open, extensively layered and highly reconfigurable. Each layer must provide concepts and abstractions in a way that don't restrict any other. Tuning mechanisms must be added as configuration/reconfiguration tools at the usage level as well as at programming level.

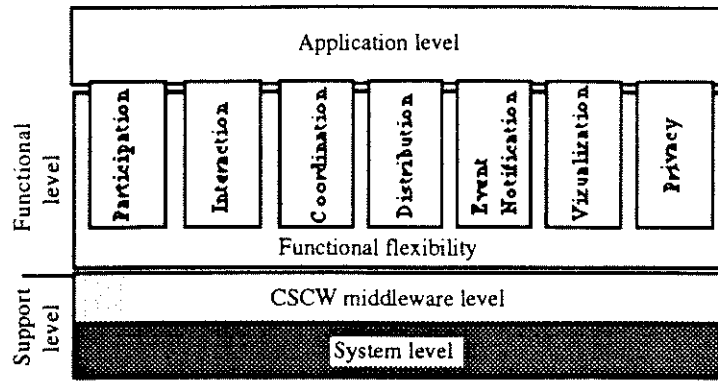

Fig.2 Application and functional criteria and related requirements

We call *functional flexibility* of a system to its ability to accommodate different functional level criteria. Functional flexibility is a measure in the functionality dimension and must be addressed as a general characteristic of the middleware level (figure 2).

3. A CSCW architectural support model

In this section we will consider the characteristics, properties and structure of a CSCW middleware level, using a group based approach. First we describe the interdependence of groupware functional criteria and group concepts and abstractions. Then we propose an integrated model based on those concepts to build a generic support level. To concretize the abstract group concept of CSCW, we will use a process group model to map group communications and management [Liang 90] as a system support basis for our approach.

3.1 The middleware architecture

The architecture of a CSCW middleware support level must incorporate the emerging results of recent distributed systems research. In the design of such layer, we must consider all the requirements above but also other important issues like: network delay, network latency and network loading. The CSCW middleware architecture can be centralized or distributed. Between these two extremes we can consider several hybrid implementations. Figure 3 shows the fundamental advantages and disadvantages associated with different approaches.

The centralized solution is based on a central server handling all the events related with each user input and display. Each user workstation only provides a graphical terminal and a windowing system server. To implement a centralized architecture, the client server model can be used. Because the application and sharing data are held centrally, the application management and data consistency is simplified. The interface programming is easier still with a networked window system like X Windows for example.

In the distributed solution it is possible to manage different copies of the program or data in each user workstation. With a fully replicated architecture, the problem of maintaining exact copies, or replicas of the application and data is not easy. Locally, each replica handles the display management and data consistency. Any event that causes a change in application data, must be broadcasted (or multicasted) to all other replicas. Because the display management is local, it is easy to support different views. The customization and tailoring of each interface can benefit from the sharing policies provided by each replica, accordingly to the user's preference.

A replicated approach is the only way to deal with scalable solutions, which seems to be a future direction of cooperative environments. The major problems concern synchronization, data consistency, ordering of events and communication semantics. To deal with these difficulties, it is necessary to implement complex distributed algorithms.

The management of distributed events like: how to support changes related with the participants in a group session, how to make dynamic registrations, how to deal with faults, how to handle with errors and exceptions, etc, can be also a hard programming work.

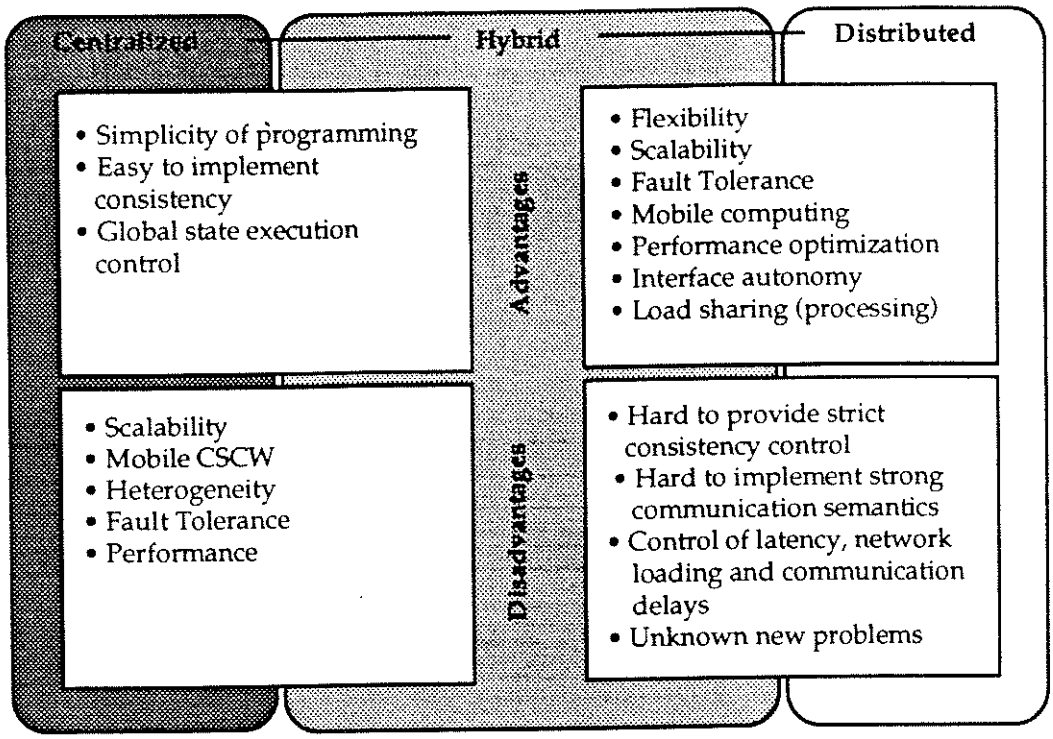

Fig.3 Centralized and distributed CSCW support architecture's

3.2 Why a group based approach ?

Each functional criteria discussed can be supported in different ways depending on the characteristics offered by a coherent integrated group based support model. A coherent and integrated group management and communication system can be provided as an operating-system-level group abstraction or at higher level. In such a system, a group is a composite of objects sharing common application semantics, as well as the same group identification (or address). A group is a single logical entity, its internal structure and interaction it's hidden to external entities.

The main motivations why objects are grouped are the following: (1) abstracting the canonical characteristics of the participants, (2) encapsulating group details and related states, (3) manipulating larger system objects as a block and (4) providing a multicast communication basis to establish different communication strategies.

The multicast communication basis subjacent to group based systems improves the efficiency of communication processes involving multiple relationships, which is essentially the case of cooperating processes. The advantages are: (1) a multicast support reduces the overhead of the sending process and optimizes the network traffic; (2) reduces the complexity of programming, if its semantics matches the application needs, and (3) hides the internal coordination of the participants involved.

We think that a group based approach is essential to build a general-purpose distributed CSCW infrastructure. There are five main reasons to justify this assumption:

(1) the group abstraction is a natural way to deal with the cooperation concept;
(2) cooperation involving many active entities can be easily described in terms of group relationships;

(3) CSCW applications will be increasingly exigent in terms of group membership management in large scale scenarios;
(4) scalable cooperative environments need (intensively) support for: multicast information channels, event ordering guaranties, replication strategies, availability, fault tolerance, dynamic reconfiguration and performance optimization. This is precisely the main motivations of group based systems;
(5) A group based approach to build CSCW infrastructures is fundamental for programming purposes. The use of programming interfaces providing group abstractions and group communication primitives (with appropriate semantics) is fundamental to help programmers to develop portable software. Using group abstractions, the programmer can be isolated from the details of group management and their complexity relationships, and can concentrate his or her efforts on each application specific needs.

There are obvious advantages in using the potential of a group oriented approach as a structuring basis to describe multiple distributed activities: performance, consistency, availability and transparency [Veríssimo 92]. In addition, standards for group communication are currently under study in the X/Open and IEEE communities. (A group oriented parallel communication standard called MPI has been introduced in 1993).

Moreover, several group oriented toolkits and environments comprising group communication protocols are now available. Several consolidated distributed operating systems also offer group communication mechanisms, in addition to conventional RPC and message streams mechanisms (Chorus [Rozier 87,90], Amoeba [Tanembaum 86,89], x-Kernel [Peterson 89,90]).

The Isis system [Birman 92] is a reference toolkit providing group communication and management. xAMP [Rodrigues 92] is another approach of a reliable group-oriented layer to structure distributed applications. Transis [Amir 92] also proposed a reliable group communication model focusing on high availability. More recently, the Horus system integrating contributions from Isis, Transis and x-Kernel [Renesse 94], emerges as a group communication system with several capabilities matching the CSCW functional requirements.

3.3 Characteristics of a group based system to support CSCW

The ideal group management layer must incorporate, with total flexibility, all the identified CSCW functional requirements. Simultaneously, key properties, accepted as relevant in a distributed strategy perspective must be preserved: openness, scalability, concurrency control, resource sharing, security mechanisms, fault tolerance, availability and dynamic reconfiguration. Flexibility is a key-criterion orthogonal to the above properties.

From the programming point of view we need to guarantee rapid and easy prototyping, reutilization of functions, optimization/tuning and facilities to accommodate different communication semantics.

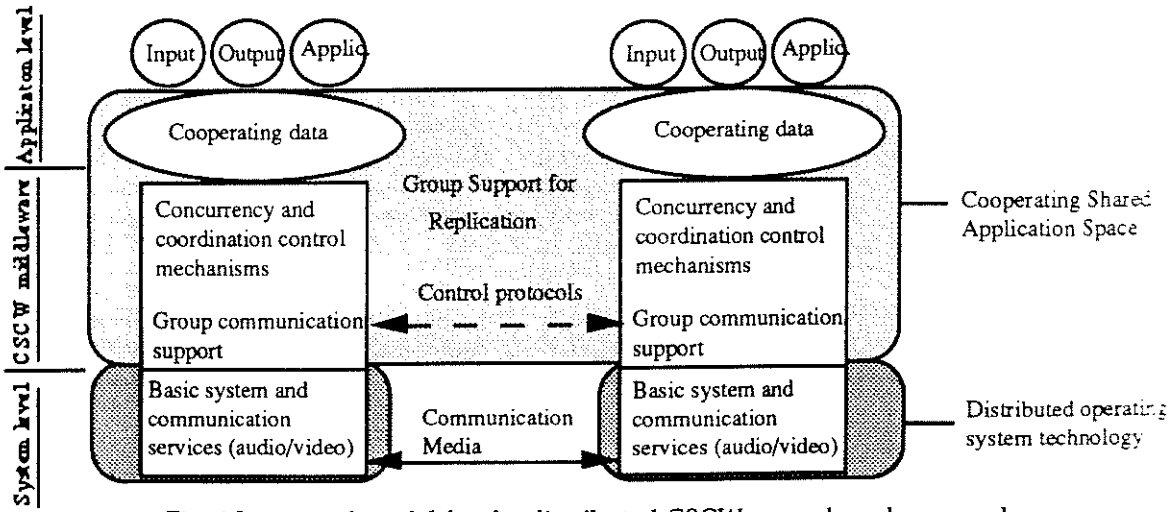

Fig. 4 Integrated model for the distributed CSCW group based approach

As we understand, the use of a group approach to structure a distributed architectural model of CSCW, can be detailed accordingly with the figure 4.

In the figure we use a distribution strategy based in group replication. Extensible group mechanisms offered by the CSCW middleware can provide the set of facilities to support the flexibility vector in the domain of space x time x functional dimensions. Centralized strategies based on a unique coordination server have a limited scalability, performance and fault-tolerance.

Each cooperating active entity is represented in the cooperating space by an agent. Using the CSCW middleware support, agents participate in one or more groups, in order to maintain a cooperative activity, globally coordinated. Furthermore, group replication techniques can be used to replicate the cooperating data component.

For the sake of openness and adaptability there are several advantages in designing the CSCW middleware as a platform providing portability. However, simultaneously, the middleware layer cannot prevent the access to eventual specific advantages of each technological support: special operating system features, new emergent transmission media, user-level/system-level context switching, etc. These specific advantages can be critical to achieve a reasonable performance. The conciliation of portability needs with those specific trends can be partially contradictory, but it is an unavoidable challenge.

3.4 Process group based support for CSCW functional requirements

In this section, we stress how process group based systems can address the trend of supporting the multiple CSCW functional requirements.

Participation: relates with the possible group organization and configuration as well as membership process management. Groups can be closed or open and must be configurable in a static or dynamic way. Groups must provide the possibility of maintaining multiple objects belonging to different groups. Distributed processes cooperating in the context of a complex CSCW application, may have to belong to different groups. In addition, groups can overlap.

Coordination: Involves group activity management and concurrency control mechanisms between participants. Fine-grained concurrency control mechanisms are needed to support different distributed shareable objects: machines, users, files, file chunks, data-structures and bytes. Concurrency control mechanisms must provide the ability to manage locks to allow the control of all cooperation activities. This control can be established either by the application core or by the user process of interaction. The system must provide fine grain locking as a basic mechanism to provide the necessary basis for distributed and sharable objects. This level of granularity, can be provided at the middleware level, on top of the basic process group support.

Distribution: Distribution is concerned with group communication and involves support for replication strategies and consistency criteria. Replication is important as a distribution strategy, but also has advantages in fault tolerance, performance and network traffic optimization. Structural classification of groups is directly concerned with different strategies of replication and consistency criteria. Each group offers a major or minor degree of homogeneity in terms of operations available and data manipulated by those operations. Different applications have different structural needs in terms of several levels of participation. The group management layer must provide different levels of consistency criteria: strong and interactive consistency, as well as relaxed consistency, or simple breakpoint mechanisms.

Interaction: Interaction deals with information dissemination, communication primitives and the establishment of interaction at different levels: peer-to-peer, intragroup, intergroup and group to group. In fact, the communication support must provide simple peer-to-peer communication but also multicast communication in a reliable basis. Reliability is related to the guarantees associated with the delivery, in a correct way, of all generated messages (even in presence of failures)[Birman 87a]. The reliability issue is fundamental to address scalability, availability and fault-tolerance. These properties must be present if we consider the development of scalable CSCW applications.

Peer-to-peer	Unicast message passing. Is the basis for the individual-to-individual interaction
Intragroup communication	Internal group communication mechanisms, based on multicast protocols. Is the basis for individual-to-group interaction in the context of the same group
Intergroup communication	Is the basis for interactions between an external entity and a group. Based on multicast protocols
Group to group:	Communication facility among different groups. Can be addressed as an extended intergroup communication facility, available to each participant of a given group.

Table II - Information dissemination

The establishment of ordering guarantees is also important. The group management layer must provide the following ordering strategies: FIFO - all the multicast messages sent are received by each group member in the same order as they are sent; CAUSAL - all the multicast messages sent are received by each group member in the same order they were sent, but only if they are related; TOTAL -all the group members receive all messages in the same order (but not necessarily the order they are sent). These ordering strategies are the basis to establish consistency criteria which are involved in cooperating and group activity coordination.

Group overlapping involving processes belonging to different groups is also an example of strong requirements in terms of message ordering. We can easily imagine scenarios where sub-groups of users work together under the supervision of a coordination group. We must also consider the different semantics of group communication. After sending a multicast or broadcast message, the originator can be interested in different scenarios. The following semantics can be useful to support negotiation and voting processes:

Atomic delivery: in this case the originator assumes that all the destination participants (e.g.: all the group) received the message or neither of them. This is called *atomic multicast* or *atomic broadcast*.

Quorum delivery: when a majority of participants received the message

K delivery: when at least K participants received the message

The originator may also need to control the number of replies associated with a multicast message. Different semantics are involved from this point of view, and can be associated to primitives available at group management layer:

No reply (none) : this means no guarantees, and is close to the asynchronous interaction mode:

Return after 1: Return after the reception of the first reply

Return after K: Return after the reception of K replies

All: Return after the reception of all replies

Event notification: group management of events and support for reactive response. Support for dynamic monitoring (including dynamic behaviour of applications) and execution control is also important. This is one of the most interesting issues provided by a group management layer. The provision of reactive programming strategies are particularly attractive.

Visualization: external data representation layers and mechanisms are necessary to provide the support for different data presentations. However, visualization can benefit from the interactive support protocols and coordination control, available from the group management layer as well as from the cooperating data replication.

Privacy: privacy requires group protection support mechanisms. It can be based in access control lists or capability mechanisms [Tanembaum 90] establishing different access rights. Network security must also be considered.

3.5 Structuring the CSCW middleware support

Many distributed operating systems have offered group communication mechanisms (in addition to conventional RPC and message passing). However, those system mechanisms (with the exception of Amoeba) are limited in terms of reliability and consistency criteria.

Other group communication protocols and tools have been implemented in user space, as was done in the Isis system. Lessons and remarks about these implementations [Birman 92] and other user-space group management layers refer a sub-optimal performance constraints due to three main reasons [Renesse 94]: (1) non-privileged user-level communication cannot always exploit multicast primitives available at underlying network level, (2) currently operating systems are not adequate to the buffering policies for group communication protocols, performing poorly in response to their needs and (3) the amount of context switches and cross-address space references is considerable [Renesse 89].

In terms of performance and portability, a CSCW middleware support requires (eventually) the possibility that group communication can run either in user space but must take all the possible advantages from multicast functions offered at the system level (e.g.: multicast data-link support). To fulfil this objective, the ideal would be to use group communication layers structured in a way that could be configured to reduce the complexity of the middleware layer, whenever this is possible. Another advantage is the exploitation of the new capabilities offered by the emergent transmission technologies available at system support levels.

Additionally, we believe that the internal architecture of the middleware must be extensively layered. This provides the ability of joining different components accordingly to the needs of multiple CSCW applications integrated in a given cooperating environment. This is very important if we intend to provide a lightweight context switching between different cooperating applications, as is available in the Chorus system microkernel.
Another important property to be addressed by the CSCW middleware is the possibility to compose different control protocols by means of stackable communication components.

Considering that fault-tolerance must be addressed in order to support scalable environments maintaining strongly coordination guarantees, the middleware infrastructure to support future groupware must provide strictly consistency as we find in the virtual synchrony model [Birman 87] [Birman 92].

In strongly cooperative edition environments, different views and versions (with different granularity objects) can be necessary. From the vizualization viewpoint, different users can be interested in managing a different version of the same object (or a specific attribute of that object). However, while different users are manipulating different views and are working and maintaining different versions of an object, a group must agree on a consistent interactive global view of the same object.

This requires fine-grained multiple views and atomicity criteria to agree on installing a new global group view, established from multiple changes. In despite that a distributed transactional-oriented strategy subjacent to the ACID[1] properties [Özsu, 1991], as it is used in distributed data bases, could be considered to deal with that problem, we believe that this approach would be highly restrictive.

---

[1] ACID relates with the following properties of database transations: atomicity, consistency, isolation and durability.

If we add a scale factor to that issue, an extended virtual synchrony concept as defined in [Malki 94],[Moser 94] seems to be the only alternative to deal with some exceptions like network partitions for example.

The extended virtual synchrony model allowing the same view to all group members, with a strictly interactive consistency criteria, is one of the most complex features that CSCW must address.

3.6 Implementation strategy

Among the different group management systems and toolkits, the Horus system [Renesse 94] emerges as a recent research system combining many of the capabilities useful to build the CSCW middleware infrastructure.

Horus consolidates a group communication architecture that treats different protocols as abstract data types. It provides simultaneously the ability to build, at run-time, other generic protocols using stackable simple protocol layers as a lego™. This approach (developed initially by X-Kernel [Peterson 89]) can be very attractive for the variety and multidisciplinary of CSCW requirements. We intend to use Horus as a basis of a generic CSCW middleware infrastructure, because it provides, in our opinion, a flexible way to accommodate most of the recognized functions we must provide to develop future complex groupware. Using Horus to develop some applications like "mtalk" tools or simple cooperative text and draw editors is quite a simple task.

However, in our approach, we are considering the possibility to build a special distributed coordination module providing a high level interface to support different granularity in group consistency control. This can be achieved using group protocol composition facilities and providing different classes of group semantics.

A future interesting trend is to develop this module in a manner that it could support a disconnected computing activity to support different phases of a long lasting collaborative task comprising several job cycles in discrete intervals of time. This can be the support for "homework" activities and mobile multi-user collaboration.

Another identified need is to provide an orthogonal security model to support privacy at different levels.

At programming level, we recognize currently some difficulties. Even a toolkit like HORUS that offers some important programming interfaces, like a group socket interface for example, is often at low level, in certain circumstances. We believe that future groupware applications require high level indirect mechanisms with more expressiveness. Group-object orientation involving group remote procedure calls and object group-oriented remote method calling [Maffeis 94], are probably important paradigms to be used.

4. Conclusions

The recent multidisciplinary approach of CSCW shows a variety of requirements and criteria associated to a large spectrum of groupware applications. A taxonomy of different applications and their needs, shows that the flexibility is a fundamental and orthogonal key-criterion to establish application level strategies, functional requirements and CSCW architectural models.

Strategies to design the main components at application level (input, output, application core and cooperating data), must be analyzed in a three dimension domain scenario: time x space x functionality. Functionality can be considered as a contribution of several functional requirements: participation, interaction, coordination, distribution, event notification, vizualization and privacy. Progress in any of the three dimensions of the domain scenario, will increase complexity. To build portable and adequate CSCW platforms supporting scalable applications, we must structure the support level in a manner able to accommodate the functional flexibility in any application domain scenario. This is a complex challenge.

We consider that the support level must be structured in two fundamental levels: the system support level which must be addressed from recent distributed operating systems research and the CSCW middleware level. In this paper we argue that the CSCW middleware level must be extensively layered in stackable components. providing a high degree of reconfiguration capabilities. To build such a middleware, we argue that the use of a group communication and management system approach is essential. Mechanisms like: group message passing, reliable multicast protocols, FIFO, causal and total guaranties of event ordering, replication and different consistency criteria are useful to solve some complex requirements of cooperative scenarios.

We proposed an integration model to provide the architectural basis for CSCW middleware. To implement this model we are using the HORUS system [Renesse 94], exploiting its group management features which we consider close to the identified needs. Future work, involves the validation of this approach, as well as some interesting trends to build a distributed programming group based framework. This framework must offer the necessary potential and expressiveness to develop complex and scalable groupware applications.

References

[Antunes 95] Pedro Antunes, N. Guimarães, "A Distributed Model and Architecture for Interactive Cooperation", Broadcast Technical Report Esprit Basic Research, project 6360, 1995

[Amir 92] Yair Amir, Danny Dolev, S. Kramer, D. Malki, "Transis: A Communication Subsystem for High Availability", Proceedings of the IEEE 22nd Symposium on Fault-Tolerant Computing, pp 76-84, Boston, July 1992

[Bentley 94] Richard Bentley, T. Rodden, P. Sawyer, Ian Sommerville, "Architectural support for cooperative multiuser interfaces", IEEE Computer 27 (5), pp 37-46

[Birman 91] Kenneth P. Birman, R. Cooper, B.Gleeson,"Design Alternatives for Process Group Membership and Multicast", Technical Report TR 91-1185 (Revision)- Dep. Computer Science, Cornell University, January 1991

[Birman 92] Kenneth P. Birman, "The Process Group Approach to Reliable Distributed Computing", Technical Report - Dep. Computer Science, Cornell University, March 1992

[Birman 87] Ken Birman, "Exploiting Virtual Synchrony in Distributed Systems", Technical Report 87-811, Dpt.of Computer Science, Cornell University, February 1987

[Birman 87a] Ken Birman, T. Joseph, "Reliable Communication in the Presence of Failures", ACM Transactions on Computer Systems.Vol 6 (1) pp 47-76, February 1987

[Cosquer 95] Francois J. Cosquer, Paulo J. Verissimo, "Survey of Selected Groupware Applications and Supporting Platforms", Esprit Project BR 6360 (Broadcast) Tech. Report 1995

[Greenberg 91] S. Greenberg, "Computer-Supported Cooperative Work and Groupware", Prentice-Hall Ed. 1991

[Greif 88] I. Greif, "Computer Supported Collaborative Work: A Book of Readings", pages 477-508, Morgan Kaufmann Publishers Inc.1988

[Gruddin 94] Jonathan Gruddin, "Computer-Supported Cooperative Work: History and Focus", IEEE Computer 27 (5), pp 19-26 May 1994

[Jarczyk 92] A. Jarczyk, P. Loeffler and G. Volksen, "Computer-Supported Cooperative Work: State of the Art, "Version 1.0, Siemens A.G tech. report, Munich, 1992

[Kaashoek 89] M. F. Kaashoek, Andrew S. Tanembaum, Susan Flynn-Hummel and Henri Ball, "An efficient reliable broadcast protocol", Operating Systems Review, 23 (4), pp 5-19, October 1989

[Liang 90] Luping Liang, S. Chanson, G. Neufeld, "Process Groups and Group Communications", IEEE Computer,pp 56-65, February 1990

[Maffeis 93] Silvano Maffeis, "ELECTRA: Making Distributed Programs Object-Oriented" IFI Technical Report 93.17, Dpt of Computer Science , University of Zurich, July 1993

[Malki 94] Dalia Malki, Ken Birman, Andre Shiper, Aleta Ricciardi, "Uniform Actions in Asynchronous Distributed Systems", Proceedings of the 14th ACM Symposium on Principles of Distributed Computing, San Diego, ACM SIGOPS-SIGACT, August 1994

[Malm 94] Pal S. Malm, "The unOfficial Yellow Pages of CSCW", M. Sc. Thesis appendix (ftp: gorgon.tft.tele.no), 1994

[Moser 94] L. E. Moser, Y. Amir, P. M. Melliar-Smith, D. Agarwal, "Extended Virtual Synchrony", Proceedings of the IEEE 14th International Conference on Distributed Computing Systems, pp 56-65, Poznan, Poland, June 1994

[Özsu, 1991] M.T. Özsu, P. Valduriez, "Principles of Distributed Database Systems", ed. Prentice-Hall, Inc. 1991

[Penz 93] F. Penz, P. Antunes, 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

[Peterson 89] Larry L. Peterson, N. Hutchinson, Sean O'Malley, M. Abbott, "RPC in the x-Kernel: Evaluating New Design Techniques", Proceedings of the 12th ACM Symposium on Operating Systems Principles, pp 91-101, Arizona, November 1989

[Peterson 90] L. Peterson, N.Hutchinson, Sean O'Malley, H.Rao, "The x-Kernel: A Platform for Accessing Internet Resources", IEEE Computer pp 23-33 May 1990

[Reinhard 94] Walter Reinhard, J. Schweitzer, Gerd Volksen, Michael Weber, "CSCW Tools: Concepts and Architectures", IEEE Computer 27 (5), pp 28-35 May 1994

[Renesse 89] Robert Van Renesse, J. Van Staveren, A.S.Tanembaum, "Performance of the AMoeba Distributed Operating System", Software: Pratice and Experience, Vol 19, n.3 (Março 1989), pp. 223-234

[Renesse 94] R. van Renesse, T. M. Hikey, K. P. Birman, "Design and Performance of Horus: A Lightweight Group Communications System", Technical Report 94-1442, Department of Computer Science, Cornell University, August 1994

[Rodrigues 92] L. Rodrigues, P. Veríssimo, "xAMP: A Protocol Suite for Group Communication", Proceedings of the 11th Symposium on Reliable Distributed Systems, October 1992

[Rozier 87] Marc Rozier, J.Legatheaux Martins, "The Chorus Distributed Operating System: Some Design Issues", in Distributed Operating Systems, Theory and Pratice, Ed. Y.Parker et al, Nato ASI Series, Vol. F28, Springer-Verlag,pp 261-287

[Szipersky 93] C. Szipersky, G. Ventre, "A Characterization of Multi-Party Interactive Multimedia Applications", Technical Report ICSI-University of Colorado, Berkeley, February 1993

[Rozier 90] Marc Rozier, et all., "Overview of the Chorus Distributed Operating System", Technical Report CS/TR-90-25, Chorus Systémmes, April 1990

[Stefik 87] M. Stefik and all, "Beyound the chalkboard: Computer support for collaboration and problem solving in meetings", Communications of the ACM, 30(1) pp 32-47. 1987

[Tanembaum 86] A.Tanembaum, S.Mullender, "The Design of a Capability-Based Distributed Operating System", Computer, Vol.29 n.4 (Março 1986), pp 289-300

[Tanembaum 90] A.S. Tanenbaum, R. Van Renesse, H. Van Staveren, G.J. Sharp, S. Mullender, J. Jansen, G. Van Rossum, "Experiences with the Amoeba Distributed Operating System", Communicatins of the ACM, vol.33 pp 46-63, 1990

[Veríssimo 92] Paulo Veríssimo, L. Rodrigues, "Group Orientation: A Paradigm for Distributed Systems of the Nineties", 3rd Workshop on Future Trends of Distributed Computing Systems, April 1992

[Volksen 93] G. Volksen, "Approach Strategies to Groupware", Proceedings Groupware 93 Europe Conf., 1993, pp 483-497

[Wiil 93] Uffe K. Wiil, John J. Legget, "Concurrency Control in Collaborative Hypertext Systems", Hypertext'93 Conf.Proceedings, pp 14-24, November 1993

[Williams 92] Neil Williams, Gordon Blair, "Distributed Multimedia Application Studie", Thechnical Report MPG-92-11, Lancaster University, Department of Computing, Lancaster, 1992