Departamento de Informática
Faculdade de Ciências e Tecnologia
UNIVERSIDADE NOVA DE LISBOA
2825 Monte Caparica - Portugal

**Technical Report**

DI-UNL-02/98

# Coordination Support in the DÁgora system: Collaborative sessions and flexibility issues

**Henrique João L. Domingos**
**José A. Legatheaux Martins**
**Nuno Manuel Preguiça**

*{hj, jalm, nmp}@di.fct.unl.pt*

Secção de Arquitectura e Sistemas de Computadores
Equipa do Projecto DÁgora

**URL Ref: http://dagora.di.fct.unl.pt/**

# Coordination Support in the DÁgora system: Collaborative sessions and flexibility issues

TR-DI-UNL-02-98

Henrique João L. Domingos, J.A. Legatheaux Martins, Nuno M. Preguiça
{hj,jalm,nmp}@di.fct.unl.pt

DÁgora Project
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Monte da Caparica
Telef. +351 1 295 44 64

***Abstract***

*In this report we discuss the principles, design options and implementation issues to support coordination and awareness services in the context of adaptive CSCW sessions. We analyze essential differences between the coordination support needed by these sessions when compared with the conventional coordination activity within workflow systems. Then we propose a flexible support model, which is more adequate for group-oriented collaborative work tasks developed in the context of a more complex workgroup activity.*

*The support is concerned with a set of adaptable mechanisms as well as system base components and services provided by an integrated, open and extensible support platform. These components implement a set of facilities for coordination and awareness control in the context of collaboration sessions.*

**Keywords**: *CSCW/Groupware, Coordination, Awareness, Large Scale Distributed Systems, Object Oriented Programming,*

## 1. Introduction

There are different definitions and characterizations that can be used to refer the coordination control activity. Different formalizations of the term "coordination", directly or indirectly applied to CSCW systems, can be found in different authors [Malone90][Holt88],[Wino86]. In the perspective of this paper we define *coordination* as the support for the activity of managing dependencies and possible conflicts between collaborative entities involved in common and inter-related tasks of a collaborative activity.

These entities are different when regarded at different levels of a computational system. At application-level entities relate users participating in a workgroup and interacting in the context of collaborative work sessions, represented by specific applications. At system-level, entities are associated with the different instances of computational processes running in a shared system-level coordination context implementing the notion of a collaboration workspace. The processes are associated with collaboration-aware applications adopted by the users when they interact with that workspace.

In the CSCW literature, the *coordination support* is referred with different perspectives and focus. In the case of specific computational system supports we can distinguish between two fundamental approaches: the approach of the coordination support provided in the context of conventional workflow systems and the approach related with the coordination issues in group-oriented collaboration-aware applications and systems.

In both cases we recognize the same base coordination principles as discussed by Malone and Crowstown [Malone94]. However, the essence, focus and emphasis of coordination control activities in both cases have important contextual differences. These differences relate with different requirements and design principles of the coordination support in the two appoaches.

Starting from the proposed definition of coordination and analyzing the properties provoking potential dependencies and/or conflicts, we observe that the coordination and its support in the case of classical workflow management systems and in the case of collaboration oriented workgroup sessions is essentially different. In each case there is a distinct context for the coordination activity. The different contexts need to be modeled and materialized by different mechanisms, base system components and management tools.

In conventional workflow systems, the different tasks (or processes) of a complex and/or longtime activity are modeled by flows determining in a more or less strict way the coordination methodology to be applied. Each task corresponds to a process scheduled by the workflow. The participants in each process, their roles, the resources to be used, the goals of each task, the scheduling information and the relationships between tasks are pre-defined and rigidly assigned in the context of the workflow description. When each task begins, the above data are well-defined for coordination purposes. Thus, the coordination support can be modeled in terms of a rigid aggregation of well-known (possible repetitive) processes implementing "a priori" well-defined work methodologies.

In the case of collaboration activities, the coordination principles are motivated by other dynamic facts or work circumstances, not necessarily known in the beginning of the collaboration. The coordination activity itself is event-driven by nature. This happens because in despite that in certain moments the collaboration goal can be defined, the methodology that grants the best productivity criteria to achieve that goal is unclear. There is place for informal interactions among the participants. Furthermore, collaborative groups need mechanisms for work-facilitation, awareness support and sharing of viewpoints to promote common backgrounds. During a collaborative session, different sub-tasks need to be scheduled in a dynamic way, adapting the work circumstances to the manifestations of possible interdependencies and potential conflicts. In short the coordination support must be adaptable and based on tailorable components and services.

Despite the essential differences, in modern organizational structures we recognize possible scenarios in which the ability to switch easily and dynamically between those different coordination contexts is a very important issue. In fact, different organizations have been spurred to implement and validate new collaborative technologies to be used complementarily with conventional workflow management systems. This situation is imposed by several factors. Rising competiveness, distribution of responsibilities and decisions, fragmented markets, specialization levels emphasis on learner and more entrepreneurial behaviors are certainly among the factors requiring the collaboration of workgroups that require the adoption of those collaborative-oriented technologies.

One of the more challenging trends is the understanding about how collaborative systems will deal with new kinds of peer-coordination models and methodologies that change behavior in ways that raise productivity, creativity, motivation and economic efficiency. While existent

groupware provides interesting technical capabilities, it is the ability to transform organizational coordination behaviors, promoting common backgrounds and collaboration aware facilities that must deserve the most relevant attention from the organizational perspective.

This situation motivates the research of coordination control mechanisms promoting points of convergence between conventional workflow coordination control and the coordination needs provided by collaborative-oriented system platforms.

This convergence can be approached in two possible ways. One direction is to analyze the coordination support of future workflow management systems as a group-oriented collaborative activity itself, in a globally, scalable and decentralized perspective. This approach has been developed in recent projects in the field of workflow systems (e.g., [Santanu 97] [Sheth 97]). Another vision (perhaps not so ambitious) is to provide effective interoperability mechanisms to switch between a workflow task and generic coordination support mechanisms managing collaborative sub-tasks in the context of coordinated collaborative workgroup sessions. In this paper we take this second direction.

**Report structure.** The rest of the report is organized as follows. **Section 2** is a brief background introducing our main motivations for the study of the coordination support **Section 3** presents an applicational scenario to discuss the essential differences between coordination contexts in workflow systems and in collaborative-oriented sessions. **Section 4** proposes a coordination model for collaborative sessions and its main support components. The components are structured in an integrated and extensible platform supporting CSCW in possible large-scale settings. **Section 5** details one of the coordination components: the session event notification service as a basis for the awareness support of coordinated sessions. Finally **section 6** ends the paper with some conclusions.

## 2. Background and Motivations

During the last two years, we have been working on the specification, design and implementation of a generic, extensible and integrated platform[1] supporting CSCW/Groupware for large scale distributed settings. The platform is based on a flexible object-oriented JAVA framework. It supports the requirements for the development and operation of groupware in large-scale collaborative work settings. With this framework we have implemented several applications and tools as an initial effort to validate the base support.

The platform is organized as a set of middleware components and base support services, structured as an extensive multi-layered architecture, providing the following functionality:

- **Communication services** - support for group-oriented multi-synchronous communication (providing a flexible support for synchronous as well as asynchronous and disconnected group-oriented interaction modes). Flexibility is achieved by synchronous and asynchronous settings and semantics covered by different protocols.

- **Collaborative object-group replication support** - support for collaboration-oriented workspace services namely: object sharing based on flexible concurrency control models and consistency management. This flexibility provides the complementary reuse of the base object-replication abstractions in the context of peer-to-peer synchronous closely-coupled collaboration models (for synchronous highly-interactive groupware) and asynchronous loosely-coupled collaborative environments (for asynchronous groupware and disconnected collaborative work).

- **Data repository service** - a repository of objects, structured as a weakly replicated set of servers, allowing client caching to cope with disconnected operation. The repository is based on a read-any/write-any model of object replication. An object framework allowing flexible type specific update conflict detection and resolution completes the object-storage service.

- **Coordination support** - a set of reusable and tailorable components, anticipating different coordination behaviors and providing facilities for adaptation to specific coordination needs of different collaborative tasks. The coordination support model implements the notion of coordinated collaborative sessions as self-contained coordination units. The users participating in session workgroups use different collaboration-aware tools and applications in a complementary way. The coordination support is also related with the awareness support and interoperable facilities between each session and orthogonal standard services (ex., E-MAIL, WWW and SMS mobile channels).

The different support layers co-exist in an integrated and flexible way, but the framework preserves open and extensible characteristics, maintaining adequate levels of orthogonality. The technical implementation details of the different support layers above are explained in several publications [Dom 97],[Simão 97],[DAgora 98] [Preguiça 98].

In our current work we have selected two main goals: (1) The operation of the platform in different collaboration contexts to validate its concepts and abstractions in the support of applications running in real world environments, (2) The analysis of the adaptability and openness characteristics of the platform to support real collaboration contexts with specific coordination needs.

To achieve these goals, we need to understand and evaluate the circumstances and requirements through which the fundamental supports described are inter-related. This is important in different perspectives: from the end-usage adaptation perspective of different organizational structures, workgroups and individual users to the software engineering perspective of the platform at different support levels.

The requirements are particularly important at the coordination support level and its relations with the object-storage service. This aspect is mainly concerned with the rest of the paper.

## 3. Coordination support: the case of workflow systems and collaborative sessions

As described above, we are particularly interested in the research of the coordination support issues, mechanisms and components commonly associated to different groupware running atop of the DÁgora system.

---

[1] The platform called DÁgora has been developed in the computer science department at the FCT-UNL. References about the system and demos are available (http://dagora.di.fct.unl.pt).

We begin by presenting an applicational scenario in which the platform and its framework can be adopted. Starting from this scenario we will discuss the requirements for the coordination support as initially defined: the way to manage dependencies and solving possible conflicts in collaborative work sessions. Then, we will analyze the fundamental differences in the coordination support of workflow management systems and in the case of collaboration-oriented workgroup sessions.

**Applicational scenario**

In a certain period of time, a group of researchers and students is engaged in a common activity composed by different tasks. Tasks are concerned with the development and validation of ideas and achievement of results, during the implementation of specific components of a software system. The results will be used in different demos (public presentations), for the edition of a technical report to be published by the research institution and for an article to be submitted to an international conference. The whole activity is modeled by a workflow, describing and scheduling all the different planned tasks (as represented in the *fig. 1*).

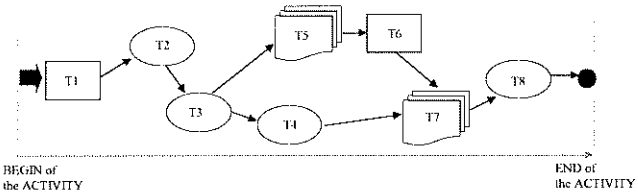


*Fig.1 The workflow description of the whole activity*

The workflow is managed by an adequate application (called a workflow manager - WFm), which is implemented in JAVA. Thus, the workflow is easily downloadable from a simple URL address by all the participants involved in the activity. The deadlines, descriptions and specific data structures related with each partial task are stored in a collaborative object-repository system. These data can be modified by simple functions at the WFm application level, by a user or users with adequate rights subject to access controls.
These workflow configuration functions are used to modify dynamically each task's attributes in the course of the whole activity. New tasks can also be added when necessary by using pre-defined task templates. These template implement base workflow components (such as: a starting point, a process, an alternate process, a decision point, a URL link to a stored document, a co-process or collaborative process, a sub-workflow or a termination point).

A task manager - Tm - allows the dynamic control of different tasks. Task managers are simple components used to manage the data types inherent to workflow base object-types (called *task templates*). In particular, a Tm can implement an instance of the WFm in a recursive way.

The WFm materializes a coordination context management application. It is a pure asynchronous application allowing different persons to manage the workflow tasks from different sites.

As represented in the figure, at certain points of the workflow, (when some expected results from previous tasks were already achieved) different co-processes are scheduled. Each coo-process corresponds to a *coordinated collaborative session*, in which several users cooperate to achieve a common goal. This goal represents the expected result of the task in the perspective of the workflow coordination context. In the collaborative sessions of our scenario, different participants will work together in a collaborative way to prepare a technical report (T5) and the article to be submitted to the journal (T7). A collaborative session is a complex task when compared with the other base-workflow tasks, which correspond to simple, well-known or repetitive work methodologies and processes (T1, T2, T3, T4, T6 and T8). For example, T8 is the task of sending the article produced in T6 by surface mail, according with the requirements of the journal editors.

The Tm for a collaborative session is called a *collaborative session manager* or *CSm*. It implements a new coordination context specifically conceived for the coordination needs of collaborative sessions. In the initial workflow, it materializes a coordination context switching between the workflow management level and the coordination level of each collaborative session.

In a collaborative session, it is not suitable to describe the internal sub-tasks as a workflow model similar to the former. The idea is that in the moment of scheduling a collaborative session, there are no precise ideas about how the collaboration must be coordinated to achieve the expectable result. At the level of the workflow, a coo-process corresponds to a point in which the workflow designer decides to propose: "lets have a meeting to deal with this new kind of task !". In our scenario, the different researchers will contribute with this perspective to prepare the technical report (T5) and the article (T7) in a peer-collaborative basis. No one has a special previous role in the context of the collaboration.

In the collaborative session, different sub-tasks will be scheduled and developed dynamically in a closely coupled way. These tasks are supported by different collaboration-oriented tools used by the participants in a complementary and combined way. In the specific present case, the users have decided to adopt a collaborative editor – which is used to develop the report and the paper, a collaborative scheduler (both are asynchronous collaboration-aware applications), a synchronous collaborative whiteboard and a video-conference tool.
By adopting the asynchronous tools, each user can work individually, possibly using her/his laptop, disconnected from others or from servers - for example a certain user is responsible in a certain moment by preparing the section 1 of the article. Gradually, the users submit their partial contributions to the session for merging. After each submission, all the users will be notified about the fact. Thus, the workgroup is permanently aware about all the individual contributions as well as eventual conflicts when those contributions were submitted (even in possible different time frames). By scheduling tasks related with he synchronous tools in adequate moments, all the users participate in synchronous meetings to prepare the next work phases. For instance a synchronous meeting was initially scheduled to discuss the outline of the article.

During the context of the collaborative session, the participants use different coordination components, managed at the level of the *CSm*, to coordinate their actions, managing dynamically the interdependencies and deciding about the remaining conflicts. With those facilities they achieve the expected results with appropriate levels of productiveness and effectiveness.
After obtaining the collaboration results, the initial workflow can proceed with the next process up to its termination.

**Supporting the applicational scenario**

In the *fig.2* the applicational scenario is represented atop of the DAgora platform [DAgora98]. The figure also presents the inter-relations between the base support levels of the platform and its framework.

In the figure we recognize the same kind of inter-relations within the framework as proposed in [Araújo 97], despite of the differences related with the motivations in each case.

The context switching between the workflow coordination level and the coordination support of the collaborative session is materialized by the *CSm*. In fact it represents an important piece to provide the adequate flexibility between the collaboration and coordination support levels of the platform [2].

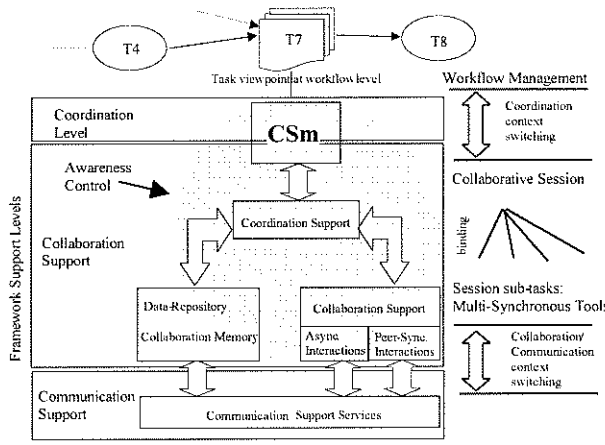


*Fig.2 The support of the applicational-scenario and the inter-relations within the framework*

As suggested by the figure, the CSm materializes a coordination context switching between the coordination support provided at the collaborative session level and the coordination and management at workflow level. It promotes a group convergent perspective about all the sub tasks developed in the context of each session. Note that each sub task corresponds to the usage of a specific collaborative application/tool. The CSm reuses the base coordination components and services provided by the coordination support level, namely: session binding facilities, user's registration services (adopted in the case of closed sessions), session sub tasks scheduling and binding and session event notification services. We will describe these base components and services later in the secion 4.

---

[2] The system also provides flexible characteristics between the collaboration and communication supports. The multisynchronous applications and tools used that can be used within a session handle replicated objects manipulated according with different semantics. The operations on these objects must be mapped in a correct way by the adequate protocols provided by the communication support services. These protocols implement different warranties (e.g., reliability and message ordering), supporting in each case the specific requirements of different object-semantics.

**Properties causing dependencies**

Starting from the initially proposed general perspective of coordination and taking a generic application scenario the next step is to characterize and understand the general properties which can cause dependencies and conflicts in collaborative sessions. In other words, we must understand the potential reasons for the need of coordination in different contexts. After this, we can discuss the common coordination components that must be present and accessible from the collaborative session manager (*CSm*) to anticipate the different coordination behaviors and adapt different needs in different coordination/collaboration contexts.

**Control of syntactic adjustments**. This property stands for the agreement on the formats of the transferred data during the group interactions and requires well-defined specifications and interoperability between the tools used to manipulate these data.

**Control of semantic adjustments**. This is enabled by the syntactic adjustments provided. After receiving the data, during the interactions, all the group members must agree about the meaning of the data. As this problem leads in part with the problem of interpreting possible implicit contents, the flexibility in integrating/using basic auxiliary communication tools (Email, video and audio tools) in the context of the tasks inherent to the session is very important. These auxiliary tools can be scheduled in the context of possible informal interactions to clarify the semantic of the interactions, helping in the case of achieve workgroup convergence viewpoints.

- **Control on information delivery**. Specified the syntactic and semantic adjustments, it is possible to support group-dissemination messages with informative contents for the collaboration. For practical issues, at least a minimum amount of time is needed to deliver and stabilize the group-disseminated messages. This requires control of message delivery. Note that this control is independent of the interaction mode supported in terms of the basic communication system (synchronous or asynchronous), because it really depends on the user perspective. The problem occurs in synchronous modes of interaction and is very relevant in large-settings, which are asynchronous by nature. In such environments, the communication delays are non-deterministic and non-time bounded. Links can fail and computers can crash. This involve other problems related with the correct sequence of the disseminated messages which can cause wrong partial interpretations with impact on coordination properties.
- **Tasks participation**. This property leads with the configurations of the internal organization structure of each collaborative group. It involves parameters used to impose (1) limits in intergroup interactivity, (2) limits in workspace fragmentation (number of sharing objects versus the number of groups involved and number of participants in each group) and (3) scheduling of sub-tasks with the definition of partial results according with the limits established in 1 and 2 and proposing adequate tools that must be used as the correct methodology for each sub-task.
- **Authorizations management.** The information disseminated by the message delivery allows acceptable levels of updating on objects shared by the group in a common workspace. Authorization control leads with the activity of introducing access rights on the operations that each participant can realize on those objects. The access-rights must be defined trough the definition of user's roles. The roles must be mapped on the information maintained by adequate registry services and authentication mechanisms.
- **Concurrency control.** Given that the authorization process describe the access rights, the concurrency control is the system-level mechanism to achieve consistency on multiple operations submitted by users acting with the same role over an object. The impact of this support on coordination issues is that in large scale environments we need to configure and support multiple models of concurrency control, from pessimistic approaches for strict models of consistency (in synchronous collaborative tasks based on applications with interfaces implementing the WYSIWIS (*what-you-see-is-what-I-see* paradigm ) to optimistic models (in loosely collaborative asynchronous interactions in a large scale workspace).
- **Control of disturbances in state transitions**. This property is important to avoid that the default system behavior can be disturbed. State transitions on objects can generate dependencies and conflicts not planned. Disturbances may arise at any time and at any location. The only way to deal with these disturbances in collaborative processes is to be aware of its manifestations. However, this awareness must be controlled and driven by each user needs in each moment. The combination of the base models of optimistic concurrency control with efficient and flexible awareness control mechanisms provide the basis for explicit human-interventions on conflict-detection and re-adaptations of coordination decisions to solve these conflicts.
- **External interdependencies.** We consider as external the properties causing dependencies and/or conflicts

motivated by manifestations of human-factors not directly related with the computational support. These factors involve sociological, psychological, anthropological and ethnographic reasons.

In a collaborative process, the above properties must be supported in a flexible and constructive way. The coordination support must promote productiveness and effectiveness criteria by an efficient customization of those properties. However, the creativeness, the potential motivation and the cooperation-awareness control of the participants must not be obstructed by restrictions imposed by the "rigidity" and "bureaucracy" of coordination policies. Thus, the coordination mechanisms and related services must be dynamically reconfigurable, complementing the social protocols with coordination information based on workgroup awareness control, explicit coordination parameters of the session and facilities provided in sharing a "group collaboration memory". The group-memory is established and gradually enriched, during each session, corresponding to the persistent storage of partial results and coordination attributed managed trough the session manager.

Thus, the emphasis of coordination is centered in the facilitation of user's interoperability, not restricting (unnecessary) their roles, not restricting specific methodologies but on contrary: promoting common backgrounds and motivating individual contributions. These aspects highlight for the need of flexible coordination mechanisms based on work facilitation strategies and session awareness control, contributing for an unified vision of the collaboration by means of a global coordination context for each workgroup activity and individual contributions.

As mentioned by [Dourish 92], the awareness control is a fundamental coordination context of understanding of the activities of the group as a way to provide sub coordination contexts in each isolated activity. This ensures that all the individual contributions can be relevant to the group's activities as a whole. Furthermore warrants an implicit evaluation of individual actions relevance with the respect to the common goals and the workgroup progress. Thus, a common session awareness support capturing different awareness information originated by complementary workgroup tasks developed in the context of that session is a fundamental issue in terms of collaborative work effectiveness and productiveness.

In large-scale settings this aspect of the integrability of awareness supports in common coordination contexts is particularly relevant because there are no obvious facilities for informal face-to-face interactions. In the other hand, workgroups are naturally dispersed, having heterogeneous backgrounds.

## 4. Implementing the coordination support

In this section we propose and discuss the mechanisms and services to support coordination in an integrated way. These mechanisms and services are associated with the coordination support level of the DAgora platform. The main aspects related with this coordination support are: (1) the definition of a collaborative-session model, (2) the flexibility provided by the object-storage to model different sessions and (3) the components and services directly implementing the system coordination support. We will describe next these three main aspects.

### 4.1 The model for collaborative sessions

Logically, a *session* embraces all tasks, procedures, data and associated users, applications and dependency relations established in the process of reaching a common goal - the session's purpose (or the session collaborative goal). Coordination among users involved in a session is used to guarantee that each participant's efforts are positive contributions to the common goal. Different types of sessions (for instance, with different objectives and different number of participants) require different coordination degrees. Moreover, in the same session, different subtasks also need different types of coordination. Consequently, flexibility and tailorability of coordination control is required. There are *open sessions* and *closed sessions* depending when the participation is free (users just binds to the session without previous authentication) or when requires the previous registration of participants (users must authenticate when join to the session).

The session represents a *coordination unity* in which all the activities inherent to each collaborative process take place.

In the framework, several properties are associated with this notion of a session: *general information* - comprising a set of attributes such as the session title, description of the session goal and associated notification services; a *workgroup of participants* - including a registry service with individual users information (as names and roles in session's context if there are roles assigned); *tasks and interdependencies* - including the information and bindings related with the tasks that will be performed (dynamically) in the context of the session.

Session's tasks have two different types: *intermediate and final*. Intermediate tasks are defined through the definition

of a new *collaborative sub-session* (the session is therefore a recursive concept). Final tasks clearly define their goals and procedures to achieve them. There are *synchronized and non-synchronized* tasks. Associated with each synchronized task it is defined (besides general task description) a set of data: its participants and associated roles if assigned (correspondent to minimal and maximal subsets of sessions participants); scheduling time; applications to be used in each task (if applicable); a reference to data produced (filled in the end of the task, by its coordinator or by some participant). Associated with each non-synchronized task it is defined a deadline for its execution and to obtain the respective data, participants and associated permissions (for manipulating the data).

The *collaborative session manager application* - CSm - controls the coordination information associated with a *collaborative session*, including all tasks executed in its context. Each participant may *jump* to some task through simple interface mechanisms provided by the session manager - applications associated with the task are started and the binding process is automatically executed (if any - used primarily in synchronoized tasks).
Although the same concepts are apparently used in both WFm (workflow management programs) and CSm (collaborative session management programs), fundamental differences exist. A workflow is used as a systematic description of well known (possible repetitive) processes to reach some goal, while the CSm is used to create and evolve dynamically and collaboratively possible unknown processes to reach the session goal.
For the above reason, flexibility and tailorability are key-properties in the context of the CSm. Not only new tasks can be created within the session, but also new types of tasks. For instance, new synchronized tasks may be defined with new different associated tools. Moreover, sessions coordinators must have the ability to modify dynamically some previously task, define it as concluded, or trace dependencies among them. These activities can be done starting from previous negotiation processes, established by previous tasks.
Although, the process to reach the session goal can be unknown "a priori", some fundamental tasks may have been identified and reused later in the context of the current or new collaborative sessions. Thus, templates for sessions exist, allowing the definition of some initial tasks. However, the overall process is unknown, and must be defined as a result of session interaction - therefore, we are not in presence of a traditional workflow. This ability to evolve session templates for collaborations done with success, reusable in the context of future activities seems to be a fundamental aspect of capturing the memory of collaborative methodologies, contributing for the widely acceptance of CSCW systems in the perspective of organizational structures.

Summarizing, the tailorability support at coordination level relates with the anticipation of coordination behaviors and the functionality aggregated in the CSm. In fact, the CSm acts as a tailoring tool to compose coordination components: the combination (binding facilities) for simple tools (Java classes implementing collaborative tasks) or complex multi-synchronous collaboration-aware applications; facilities to dynamically change the session customizations, dynamic configurations of existent tools and applications; and ways to control session-awareness notifications. All these facilities are provided as end-user level adaptations in the perspective of individual users, workgroups and organizational coordination. The system also provides facilities for reusing coordination contexts applicable to other future similar activities.

## 4.2 The Data-Repository and its flexibility

As described, the session constitutes a coordination domain in which workgroups collaborate. The notion of session is maintained persistently during the time in which the different session tasks are developed. All the coordination context of a session needs to be managed in a persistent way during that period of time. In the platform, this context is naturally preserved by the object-storage system. The tailorability needs at session coordination level are supported by flexible mechanisms provided in the object-repository. To understand the type of base support provided by this service and its flexibility properties, we will summarize its main characteristics and functionality.

The object-storage support is a replicated data repository specially designed to support collaborative applications in large-scale heterogeneous environments including mobile and disconnected computing (see the DAgora storage system (DSS) for details [Preguiça 98]). In such setting two major problems arise: data availability and concurrent updates merging. The first is tackled by the combination of weakly consistent server replication and client caching. The second, through an open object framework that enable type specific conflict detection and resolution.
DSS manages specially structured objects, known as coobjects (from collaborative objects). These coobjects are structured according to the object framework needs, and are specially designed to handle concurrent updates. Coobjects are organized in sets, known as volumes. Each coobject belongs to a single volume. There is a direct

mapping between a *coordinated collaborative session* and a volume. Thus, each volume represents a collaborative workspace, containing coobjects relative to a given session.

The object-storage system architecture is composed by a group of servers that replicate volumes with a *read any / write any* model of data access. Coobjects are modified through method invocation, and updates are propagated among servers using log-propagation and through an epidemic scheme requiring only pair-wise occasional communications [Petersen 97]. Clients cache key coobjects to allow users operation even while disconnected (and to improve performance). The overall architecture is outlined in figure 3.

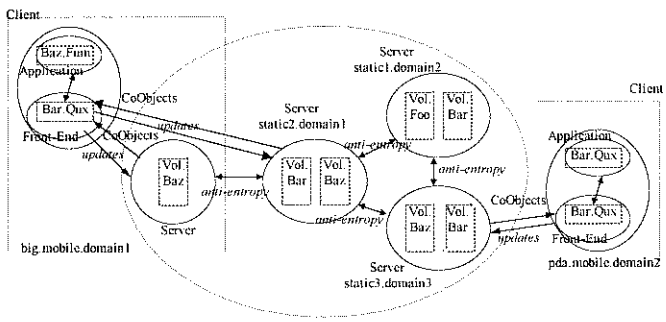


*Fig.3 – The object storage architecture.*

The service provides an open object JAVA framework that allows new data types to be composed from reusable predefined components and regular object classes, thus hiding from application programmers the complexity associated with data distribution and replication management. Coobjects are structured in several disjoint components that encapsulate different aspects of coobjects operation (see fig. 4). Different components implementations exist and new ones may be created. For instance, several policies are defined to apply concurrently made updates to different replicas, thus allowing each data type to incur only in specific overhead. This object framework allows type and situation specific conflict detection and resolution in a tailorable way.

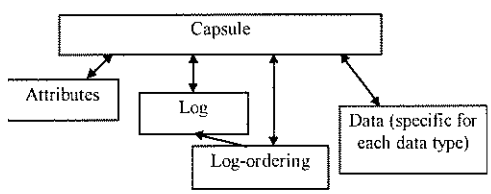


*Fig.4 - The open object DSS framework.*

Two different kinds of flexibility and tailorability are basically supported by the data storage: software engineering tailorability and system level tailorability. Next, we briefly discuss the existent mechanisms to support those different levels of tailorability.

**The programming support and aspects of flexibility**

In DSS, flexibility is achieved through an open object framework. Different semantics may be imposed to different coobject based on the same class definition. For instance, from a simple scheduler class definition, two distinct coobjects may be created: 1 – a coobject that transparently stores two versions of the data stored in the scheduler class (a tentative and a committed version [Terry 95]) and uses a sequencer for committing updates; 2 – a coobject that has a single version of the scheduler's data and commits updates through a total order based on stability techniques and uses undo/redo techniques.
This framework allows inexperienced programmers to create coobjects relying on predefined components to impose consistency among replicas, thus hiding the complexity associated with data distribution. New components, with different semantics, may be implemented, as required for new applications.

DSS allows adaptation to different and modifiable environment conditions. We present the existent flexibility mechanisms.

**Different Client Caching and Communication Policies.** The client component of DSS allows several policy modules to be specified, namely: *communicational policies* allowing different types of connections, protocols and network resources usage; *cache and pre-fetching policies*, thus allowing adaptation to different professional users (it as been verified that different users professionals have different access patterns [Pang 92]) and *different computer types and connectivity quality of service* (static, mobile with good wireless connectivity, primarily disconnected mobile, etc.).

**Different Epidemic Synchronization Policies.** The service allows a *per volume definition of epidemic synchronization policy*, through definition of (interrelated) topology and frequency of *anti-entropy* sessions. Flexibility in definition of topology allows, for instance, efficient use of communicational resources through mapping of *anti-entropy* topology to existent physical infrastructure. For example, the right-most topology *fig.5*, may be used in some collaborative project between two distinct institutions – servers inside each institution establish sessions with each other and a single connection links the two institutions.

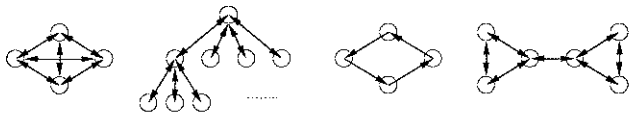

*Fig.5 –Example topologies for anti-entropy sessions that can be supported*

Frequency and selection of time periods for *anti-entropy* sessions may also be defined, making a trade-off between server consistency and communicational resources usage. For the same volume, different *anti-entropy* links may have different frequencies. For instance, in the above example, servers in the same institution may synchronize among themselves several times per hour, while the anti-entropy session between the two institutions may be executed only once a day (perhaps when communicational costs are lower).

**Different Connections and Protocols.** In large-scale and mobile settings, it is important the use of alternative communicational resources (different connections and protocols) in order to provide system adaptation to different environments. Variable connectivity should also be tackled. In the object storage system level, we provide this flexibility both in client/server and in server/server communications.

**Different Structural Organization.** In a large-scale environment including mobile computers, different computers with different hardware and communicational resources coexist. Thus, different configurations must coexist. In DSS, multiple computer configurations are possible combining server and client components in the same static or mobile computer. In *fig.6* we represent an example of a large-scale DSS setting, where multiple configurations coexist.

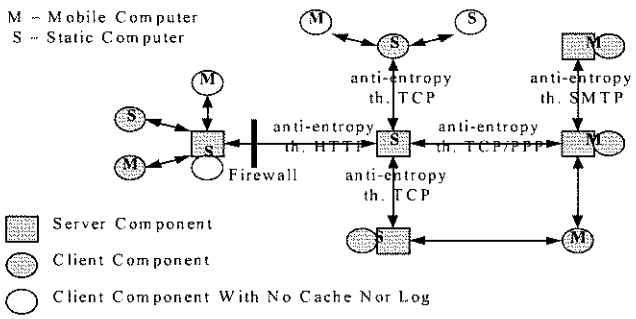


*Fig.6 – A typical large-scale environment supported*

## 4.3 Session coordination components and services

The session services implementing the coordination support level are: the session binding service, the user's registration service, the tasks scheduler and its multi-tool binding components and the session notification service.

All these services are accessible from the *CSm* environment conceived an application supported in base components interacting with all the above services.

We will describe briefly each one of those services explaining in more detail the implementation of the awareness support provided by the session notification service.

### 4.3.1 Session binding

The session binding service is responsible by providing browsing and binding facilities to all the sessions supported in each moment in the data repository. Each session has an unique name (similar to an usual web URL address, ex:
http://dagora.di.fct.unl.pt/SessionID).
The sessionID is an unique identifier. Each session maps on a volume as explained above when we described the data-storage component. Sessions can be registered in a session name service, with the set of coordination attributes. A user binds to a session just by downloading initially a CSm which provides the adequate functions to access to the coordination context of the session (coordination attributes related with the session model explained above and facilities for all the other session services).

### 4.3.2 User's registration service

In the case of closed sessions, when the session is created and configured, the users that will participate in the tasks of the session must be registered. This is done by means of the user's registration service. To each user is assigned a session role: coordinator, participant or observer. These roles are used by the session manager to provide access control rights in the context of the session. Coordinators can modify the coordination attributes in the context of a session (modifying for instance the roles assigned to the users) and can also schedule new tasks choosing the adequate tools (using the scheduler component of the CSm). Participants act as users without privileges to modify the coordination context but they can use the different tools related with the tasks scheduled and announced in the session manager. With these tools they collaborate to achieve the different goals subjacent to the scheduled tasks. When using the different tools they can handle the different object-types managed in the application-specific context. Finally, observers only use

the session manager to bind to tasks results (they cannot bind to the applications concerned with those different tasks). A result is a static information accessible by a link managed in the context of the session manager and represents a snapshot or annotation of a coordinator user (reporting the state or the result of a previous task).

#### 4.3.3 Tasks scheduling and multi-tool binding

This service provides the way to schedule and announce new tasks in the context of a session. Each announcement has all the necessary binding information to users automatically bind to each task. When a task is created (by an user acting as coordinator) it is chosen the tool that will be used to work cooperatively to develop that task.

#### 4.3.4 Awareness support

The awareness support is provided by a service called Session Events Notification Service. This service provides a generic way to disseminate and notify events occurring in the context of all the tasks developed within a session. We explain the architecture and functionality inherent to this service in the next section.

## 5. Coordination and Awareness Support: the Session Events Notification Service

The session coordination support provides an abstraction for the notification of awareness data originated from events occurred in the context of different collaborative tools used in a collaboration session. The support is provided in an integrated way by a service called the DÁgora Session Events Notification Service.

As suggested in the fig.7, the service follows a publishing/subscription methodology acting as a tailorable component between event suppliers (the collaborative applications) and the event consumers (notification session objects).

All the asynchronous collaborative tools running in the context of a session act as potential event sources, publishing different events in a set of awareness channels allocated for that session. These events correspond to the operations (marked as *loggable*) executed in the context of such applications. These operations are associated with the submission of work (coo-objects and respective data) to the object-storage service. Thus, when a user decides to submit a *loggable* contribution in the context of a session, this contribution is immediately notified through the session notification service, originating awareness events that will be subscribed in a selective way by a set of special notification objects managed within the respective session.

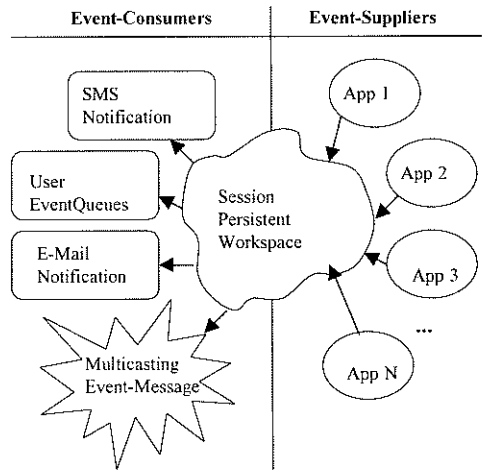


*Fig.7 Overview of the Session Notification Service*

The potential event-suppliers that can generate event feeds to be subscribed by the notification objects are related with asynchronous collaborative applications. Synchronous applications can also generate events but in this case these events only relates with two situations: when a user binds to the respective synchronous task (starting an instance of the application) and when a user ends the task (closing the respective synchronous application). Note that during synchronous interactions, the event notifications correspond to the management of active replicated objects handled in the context of each application usage. Thus, when users are in synchronous tasks, they are aware in "real-time" and in a "peer-basis" about the operations done by the other users present in the task.

The event feeds related with the whole activity developed in the context of a session is supported by a channels model as explained below.

**Event Channels**. An event channel acts as both an event's consumer channel (in the publishing perspective of event generators) and an event's supplier channel (in the subscribing perspective of the event consumers). In our model of awareness channels the communication between publishers and subscribers is completely decoupled and implemented trough interoperable object requests between them. Each event has specific data associated with each kind of request attributes. The requests depend on the basic type of the operation: subscription operations (operations instantiated under the initiative of the

potential consumers) and publishing operations (operations instantiated by the initiative of event suppliers).

At the channel level, events correspond to untyped objects and are opaque. Event source publishers and event consumers manage these events has typed-objects handled at each specific level.

In a subscription operation type, each consumer adopts an interface as follow:

```
Interface EventSubscription {
    Void event_subscribing
    (channel descriptor, request data);
    Void stop_subscription();
};
```

Each consumer initiates the interaction with a session awareness channel by invoking the event_subscribing operation on the session channel and just pushing all the events correspondent to different loggable objects stored by the applications in the data-repository service. The stop_subscribing operation can be used to terminate the reception of events.

In the case of the publishing operation type, the event suppliers adopt an API as follows:

```
Interface Event_Publishing {
  Any_data Publish (event_data);
  Void stop_publishing(channel descriptor);
}
```

The combination of the two APIs provides great flexibility in dealing with different requirements related with possible coordination criteria: the subscription operation does not have to actively solicit events and requires no buffering of events received, as the awareness data arrives as a continuous feed. The publishing operation has the advantage that it is possible to control over when each event will be delivered and it is also possible to materialize specific buffering policies at the level of the awareness channel implementation and each application-specific context.

**Session Notification Objects.** These are the objects used to capture events in a selective way. There are four types of notification objects in the session: SMS, E-MAIL, POP Event Boxes (all used as asynchronous notificators) and the Synchronous Event Multicaster Agent (used as a synchronous notification mechanism that multicasts messages containing different events to a multicast address registered to each session). Notification objects correspond to reusable components that can be used in a generic way: as components of the session manager applciation (CSm), as components of any collaborative application or in the context of simple and isolated tools. To each component is associated a channel subscription interface of a channel (representing the input interface) and a consumer endpoint (representing the output interface). Each notification object has a specialized endpoint, according with its purpose:

- SMS - an IP address and TCP-port of an SMS gateway. Events are notified by sending SMS messages to the user's pagers or phone numbers, through the interaction between that gateway and the SMS service provider of each user.

- MAIL - A simple E-Mail address. Events are notified as E-Mail structured messages supported on the SMTP protocol.

- User Event Queues – Each session maintains User Event Queues where the events are logged. The events are captured asynchronously under the initiative of users. For this purpose, a POP protocol (similar to the POP MAIL protocol) is used to capture the events from the event-queues. To do this, a special EVENT-POP component can be used.

- SEMA – a session event multicaster agent that disseminate events synchronously (by multicasting messages). Each current session allocates an IP multicast address for this purpose. This notification object acts as a radio broadcast channel, disseminating all the events in a continuous way for all the users running an instance of the session manager, when they are "logged" in a session.

**Types of Channels**

There are four types of channels in the context of the session event notification service. ESC – Event Source Channels, UC – User Channels, SAC – Session Awareness Channel and F – Filters.

**Event Source Channels**. These channels are allocated by all the instances of collaborative applications running in the context of a session. Each instance corresponds to a specific user. We explain below how these applications notify their events by send event-messages to the respective ESC. Thus, the suppliers of ESCs are the collaborative applications used in the conetxt of sub-tasks

of a session. The consumers of the ESCs are the user-channels.

**User Channels**. A user channel multiplexes the events originated by all the applications running in the session by the same user. Thus, we have an user channel per user in a session. The events of UCs are subscribed by the SAC – session awareness channel.

**Session Awareness Channel**
The SAC multiplexes all the events originated by the UCs. Each session has only one SAC. A SAC is subscribed by any of the notification objects described above.

As all the potential session notification objects may be interested in receiving only specific events, to reduce the disturbance in the user-level perspective, it is necessary that these objects can configure the related endpoints in the session awareness channel subscribing interface. This functionality is provided by the use of specific channels designated as filters.

**Filters.** The filters act as special awareness channels, in which all the events that don't satisfy a certain condition are simply discarded. To do this, the filters need to evaluate the event data by matching specific attributes, which are configured dynamically by tailorable functions of the notification objects. It is simple to implement filters because they are just subscription awareness channels with only one notification endpoint and one subscribing interface. Note that there is no need to store or retry to supply specific events. In case of eventual difficulties, a filter just can abort the notification.

The awareness channels model implementing the session event notification service is represented in the fig. 9.

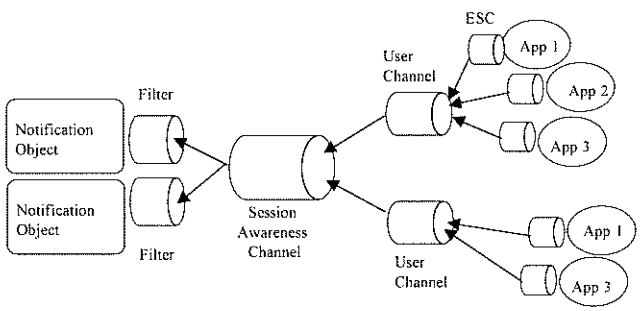


*Fig. 9 Awareness Channels Model*

The filters are interposed between the session awareness channel and each session notification component (managed in the context of a specific application - for example the session manage)r. In this interposition the awareness filter is logically implemented as a thin layer atop of the data-storage, which maintains all the persistent data of each session organized in a replicated volume, including the management of all the potential *loggable* objects managed by each application.

**Integrating Notifications in the Data-Repository**

In asynchronous applications, users collaborate through concurrent modifications to the same data (coobjects). As users modify coobjects through methods invocations, notification messages are intimately related with these invocations. As it has already been referred, coobjects are structured according to the DAgora object framework. To ease integration of notifications in DAgora system and to alleviate programmers from dealing with its associated details, we have devised a linguistic support to enable easy association of notification messages to *write* operations (operations that modify coobject state).

```
public class SchedulerCapsule
        extends dagora.dscs.TwoVersionsCapsule
        implements java.io.Serializable
{
    public SchedulerCapsule() {
        attrib=new dagora.dscs.AttribSet();
        logcore=new dagora.dscs.LogCoreSeqImpl();
        commitData=new SchedulerData();
        commitLogOrder=new dagora.dscs.LogTotalSeqCausal( false);
        tentativeData=new SchedulerData();
        tentativeLogOrder=new dagora.dscs.LogNoOrder( true);
    }
}

public class SchedulerData
        extends dagora.dscs.DagoraData
        implements java.io.Serializable
{
    public Vector appointments( int year, int month, int day) {
        /* method code here */
    }

    public loggable void insertReservation( ReservationEntry[] altRes) {
        /* method code here*/
    }
    notify {
        return new SchedulerEvent( "New reservation"+ altRes.toString());
        /* notification message to be submitted to the session
           context by means of the
           session channel notification service */
    }
    public loggable void removeReservation( ReservationEntry res) {
        /* method code here */
    }
    notify {
        return new SchedulerEvent( "Reservation removed"+ res.toString());
    }
}
```

*Fig.10. Scheduler coobject implementation.*

*Fig.10*, outlines the definition of a scheduler *coobject* used in an application we have developed. The

*SchedulerCapsule* defines the composition of the coobject, allowing not only data replication to be handled with the desired semantics using predefined components, but also a two version data coobject to be defined independently of the data object definition (for more details, see [Preguiça 98]. In *SchedulerData*, methods that modify object state are qualified as *loggable*. In clients, invocations of these methods are transparently logged, and are later uploaded to a server.
When servers receive operations performed by users in some client machine, they automatically publish the events associated with the methods performed on the specific source-channel. Notification messages associated with each method are defined in its *notify* construction. Coobjects definitions are preprocessed to generate *standard* Java code, which is later compiled using standard development tools. Thus, details involved in publishing a notification, such as finding the adequate source channel relying on user and session information are hidden from programmers, allowing easy integration of notifications in the data-repository. These notifications are then recuperated by the adequate object-notification objects according with the users preferences defined in the session context.

## 6. Conclusions

In the paper we analyze the essential differences between the coordination support provided by conventional workflow management systems and the case of the coordination support for collaborative sessions. The main motivation for this analysis is based on the background and experience in materializing a generic, flexible and integrated CSCW platform and framework for the requirements of groupware running in large scale settings.

In such a platform, the main support components are structured as an extensible middleware architecture providing base abstractions and support services at different levels, namely: group-oriented communication services, collaboration support, data-repository service and the coordination support.

The coordination support is composed by a set of coordination components and services organized in the context of a fundamental notion of collaborative session.

To discuss the requirements related with the coordination support, we present an applicational scenario in which the above platform can be used as a generic support. From this scenario, we discuss the essential differences between the coordination contexts in the case of workflow management systems and in the case of collaborative-oriented sessions. In this discussion, we characterize some properties causing interdependencies and conflicts in collaborative activities that need to be managed in the context of a flexible coordination support.

Finally, we propose a coordination model for collaborative sessions describing the main support components. In particular we detail the case of the session event notification service, that constitutes one of the main coordination components related with the awareness support, in the context of collaborative workgroup sessions.

## Bibliography


[Araújo 97] Renata M. Araújo, Márcio de S. Dias and Marcos R. da Silva Borges, "A Framework for the Classification of Computer Supported Collaborative Design Approaches", in Proc. of the CRIWG'97 - 3rd CYTED RITOS International Workshop on Groupware, pp 91-100, October 1997

[DÁgora 98] References and demos about the DÁgora project: http://dagora.di.fct.unl.pt

[Dom 97] Henrique J. Domingos, J. Legatheaux Martins, Nuno M. Preguiça and Jorge F.Simão, "Support for Coordination and Flexible Sunchronicity in Large Scale CSCW" , in Proc. of the CRIWG'97 - 3rd CYTED RITOS International Workshop on Groupware, pp 81-90, October 1997

[Dourish 92] Paul Dourish, Victoria Belloti, "Awareness and Coordination in Shared Workspaces", in Proc. of the ACM CSCW 92, pp 107-114, November 1992

[Holt 88] A. Holt, "Diplans: A New Langiage for the Study and Implementation of Coordination", ACM Transactions on Office Information Systems, n. 6, vol.2, pp 109-125, 1988

[Malone 90] T. W. Malone, K. Crostown, "What is Coordination Theory and How Can It Helps Design Coooperative Systems", ACM CSCW 90, in Readings in "Groupware and Computer-Supported Cooperative Work", Writen and Edited by Ronald Baecker, pp. 375-388, Morgan Kaufmann Publishers, 1993

[Malone 94] T. W. Malone, K. Crostown, "The interdisciplinary study of coordination", ACM Computing Surbeys, 26, pp 87-119, Mar 1994

[Pang 92] J. Pang, D. Gill, S. Zhou. "Implementation and Performance of Cluster-Based File Replication in Large-Scale Distributed Systems", *Technical Report*, Computer Science Research Institute, University of Toronto. August 1992.

[Petersen97] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, A. Demers, "Flexible Update Propagation for Weakly Consistent Replication". In *Proceedings of the 16th ACM Symposium on Operating Systems Principles,* 1997.

[Preguiça 98] N. Preguiça, J. Legatheaux Martins, H. Domingos, J. Simão, "System Support for Large-Scale Collaborative Applications" Technical Report, TR-01-98 DI-FCT-UNL, Dep. of Computer Science, FCT - New University of Lisbon

[Santanu 97] Paul Santana, Edwin Park and Jarir Chaar, "RainMan: A Workflow System for the Internet", in Proc. of the USENIX Symposium on Internet Technologies and Systems, pp. 159-179, December 1997

[Sheth 97] Amit Shet and K. Kochut, "Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems", TR

[Simão 97] Jorge F. Simão, N. Preguiça, J. Legatheaux Martins and Henrique J. Domingos, "DÁgora: An Object-Oriented Groupware Platform", Workshop on Groupware Platforms, ECSCW 97,

[Terry 95] D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, C. Hauser., "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System". In *Proceedings of the 15th ACM Symposium on Operating Systems Principles,* December 1995.

[Wino 86] T. Winograd, F. Flores, "Understanding Computers and Cognition: A New Foundation for Design", Norwood, NJ Ablex, 1986