Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work,
September 10–14, Stockholm, Sweden
H. Marmolin, Y. Sundblad, and K. Schmidt (Editors)

Supporting Cooperative Awareness with Local Event Mechanisms : The GroupDesk System

Ludwin Fuchs* , Uta Pankoke-Babatz, Wolfgang Prinz*
GMD – German National Research Center for Information Technology
Institute for Applied Information Technology
Schloß Birlinghoven
D-53731 Sankt Augustin , Germany
e-mail: ludwin.fuchs@gmd.de, uta.pankoke@gmd.de, wolfgang prinz@gmd.de

Abstract

An event distribution model for a computer based cooperative working environment is presented. The proposed model aims to provide information about the ongoing and past activities of collaborating users, based on the semantics and contextual relationships of the shared artifacts and contributes to increase the awareness of the ongoing state of affairs without overloading the user with additional information.

GroupDesk, a prototype implementation of this model is introduced. The system provides a simple environment for the coordination of cooperative document production. Support for shared awareness is achieved by visualizing the event information using the desktop metaphor.

* This work has been supported by the European ESPRIT Basic Reasearch project COMIC (ESPRIT BR 6225)

1. Introduction

In the CSCW community the problem of supporting shared awareness among the users of systems for the support of cooperative work has gained much attention amongst researchers and is discussed quite controversially (Dourish and Bellotti 1992; Fuchs, Pankoke-Babatz et al. 1994; Pankoke-Babatz 1994; Sohlenkamp and Chwelos 1994). The discussion is motivated by two issues: on the one hand the problem of making the currently ongoing activities of interest visible to the users of the system and on the other hand to provide an overview about changes in the past concerning the objects of work.

Approaches to solve these problems differ very much in their respective orientation. They range from systems settled in traditional database technology, such as version and configuration management systems (Dittrich 1986; Belkathir and Estublier 1987; Kaiser and Perry 1987) to multimedia based information systems (Streitz 1992) or three dimensional virtual worlds (Benford and Fahlén 1993). All these systems have in common, that they focus on just one of the sub problems mentioned above. As an example, the spatial metaphor of Benford and Fahlén (Benford and Fahlén 1993) has proven to be especially suited to provide an awareness in synchronous cooperation and to support guidance of synchronous communication in potentially dense populated spaces, whereas the visibility of asynchronous changes seems to be more problematic to achieve. Conversely the traditional work on configuration management aims at object consistency in asynchronous work situations.

In this paper we present some ideas to enable an integrated description of the state of cooperation. Instead of conceptually separating the actors from the objects of work the model integrates the users, work artifacts, tools and resources, into a common organizational context and allows the provision of information concerning synchronous as well as asynchronous situations The model is based on the representation of the working context as a semantic net. The nodes of the net represent the work artifacts, the actors (users), and organizational entities, such as departments, roles and procedures. The edges of the net are formed by different typed relations. Such a relation may describe similarities of artifacts in terms of content, or they can describe currently ongoing activities in the environment. They are also used to embed objects into the organizational context. The net is formed and continuously modified by the normal interaction of the users with the system.

A flexible event distribution strategy is applied, which distributes the events based on the user's interest in work situations. Users may get informed dynamically about events, that happen currently or that have happened in the past in the surroundings of their actual position in the work environment. This strategy has the advantage, that the visibility of events is bound to the user's current work occupation. Hence, the model provides a conceptual approach to prevent informa-

tion overload. It allows the support for orientation in a very general sense: information about events is not only present at the directly involved objects, but also at objects that are related to them in some specific way. For modifications this behavior plays an important role, since the state of artifacts often cannot be determined clearly in isolation from related objects.

In the first part of the paper, we outline, which kinds of awareness the event model is capable to support. This is followed by a description of the representation of the work setting. We present the core event propagation mechanism and show how it uses this representation. and how it provides the necessary information, to support the respective modes of awareness in these situations.

In the second part of the paper, we introduce the GroupDesk system, a first prototype implementation of the event model, and show how the event related facilities of the system make use of these concepts and enable an implicit awareness of the users about the overall dynamics and state of work

2. Modes of Awareness

Orientation in cooperative processes is based on events in these processes. In the following we use a notion of events, that allows a description of the state of cooperative situations and is suited to provide information to support each of the different modes of awareness, presented in Figure 1.

Synchronous awareness is concerned with events, that are currently happening, whereas asynchronous awareness considers events, that have occurred at some time in the past. Support for the latter mode needs to be derived by a summarizing interpretation of a whole sequence of events, that have happened in the meantime. Synchronous awareness should be supported by an immediate reflection of the ongoing affairs at the graphical user interface of the system.

	synchronous	**asynchronous**
coupled	what is currently happening in the actual scope of work ?	what has changed in the actual scope of work since last access?
uncoupled	What happens currently anywhere else of importance ?	Anything of interest happened recently somewhere else ?

Figure 1 Modes of awareness

Orthogonal to this classification we distinguish according to the current interest of the user between coupled and uncoupled awareness. Coupled awareness denotes the kind of overview, that is closely related to the current occupation of the user. An example for this kind of orientation is the knowledge of a user, who

wants to edit a certain document, that this document is currently read by someone else. With asynchronous coupled awareness we mean situations, when a user is working on a certain object and gets informed about changes, that happened to this object in the past during a period of absence.

Uncoupled awareness applies in situations where information about events needs to be provided independent of the user's current focus of work. As an example for uncoupled asynchronous awareness consider a situation where a work flow system sends an object, such as a spreadsheet or a folder of documents to be worked over, to somebody who s currently on holidays. If there is a deadline attached to it, then it may be very important to notify the initiator of the work flow about this – even if he is at the moment concerned with something else.

3. The GroupDesk Model of a Working Environment

3.1. Objects

The basic units of information in the system are objects. Work artifacts in the environment, such as documents, tools or working resources of any kind, are modeled as respective objects. The same holds for more abstract entities, that compose the organizational context of work: groups, departments, organizational roles and rules are all simply objects in the system. Furthermore, we integrate objects that represent the users of the system. In terms of the model, they are basically treated in the same way as any other entity the system manages. In the following, we will however refer to objects representing users by the term actor, to distinguish them from the other objects in the system.

3.2. Relations

Relations are used to place the actors and artifact-objects into a collaborative context. Relations are typed and may be grouped into three basic categories: *structural* , *operational*, and *semantic relations*.

Structural relations are used to describe any kind of relationship between objects and an associated organizational context. Examples are all kinds of membership of entities and actors in specific contexts, such as projects and departments. Operational relations are always relations between an actor and an object. The general semantics of these relations is the fact, that the corresponding actor is currently involved in some kind of activity concerning the destination object. In an environment for document production, we would e.g. express the fact, that a user is editing a document by a corresponding operational relation. Semantic relations are used to express any semantic similarity between two entities in the system. They are highly dependent on the concrete nature of work to be performed.

The general form of the overall representation spans a semantic network. The actual maintenance and evolution of the network is triggered by the interaction of the users with the system: users may create objects and move them around as they like. The system performs the insertion and removal of the required relations. Also, the establishment of operational relations is derived automatically by the system, according to the actions, the users are performing. The system reflects the dynamics of the actions because the relations are only valid during the time the activity is happening. In many cases it is also possible to derive semantic relations by the system, e.g. a versioning system could introduce specific similarity relations between different versions of design objects.

3.3. Events

We distinguish two basic types of events: modifications and activities. Modification events are generated by the system, each time the state of an object changes due to some action of a user. Activities describe synchronous events, related to the users in the system. Their creation marks the starting point and their deletion the end point of the corresponding action. Here we may imagine events such as usage of tools, presence of a user in a certain working context or synchronous communication. Of course, this list is not complete. We can basically imagine any kind of event, that has a certain relevance when it comes to coordinating the work in a given setting.

Similar to the object class hierarchy there exists a class hierarchy for the events as well as for the relations. Furthermore there is a mapping between classes in the object class tree and classes in the event and relation hierarchy, in the sense that a particular class of objects may raise a particular set of events and can establish a well defined set of relations to other objects. This mapping is "inherited" to subclasses, but may be more specific as is illustrated in Figure 2. The vertical lines indicate the baseclass - subclass relationship.

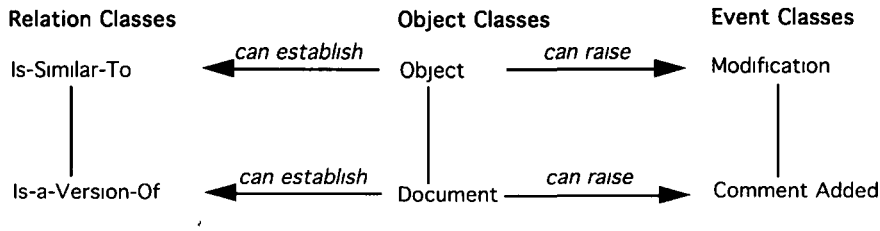

Figure 2: Relationship between Objects, Events and Relations

4. Awareness in Work Situations

4.1. Work Situations

A central requirement for the provision of awareness is to allow users to determine what they are interested in and what they are not. Thus notification of awareness information should not be prescribed by formal work representation. On the contrary the user should not be forced, to continuously register his interest for each and every object. So the system needs to offer a notion of work situation as a means to specify interest, such that each time the user is involved in one of these situations, he receives the awareness information, he is interested in.

Following the design rationale presented in the preceding chapters, we may consider a work situation for a given user a set of objects, interrelated in some specific way. An actor is involved in this situation, if one of the objects is interrelated to the actor by at least one relation. A simple example is the situation "working in a shared workspace" as illustrated in Figure 3: the situation consists of all objects that make up the workspace and the actor is involved in the situation until there is no longer some relationship between the actor an any of these objects.

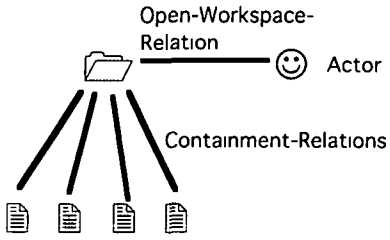

Figure 3 A work situation

4.2. Interest Contexts

Work situations form a suitable metaphor for the user to specify his interest in events. Interest in events for such situations is defined by interest contexts, which consist of a set of relation types, a set of event types and a list of interested users who have subscribed to the context. For any given object class in the system the user may define and/or subscribe to an interest context. The semantics of an interest context is, that the system maintains events of the indicated type raised by an object of this class in the surroundings of the object. The surroundings define a working situation and consist of all objects, that are linked to the original object by relations of the types listed in the context description. An example of an inter-

est context for the class document involving the event "document modified", is shown in Figure 4.

This interest context is defined for situations where the subscribing user is the owner of the document. in which case he gets a synchronous awareness about all changes of documents he owns.

The concept of interest contexts can be fully integrated into the object oriented modeling paradigm. Each class in the system inherits the interest contexts of it's parent class. Furthermore users can override their subscription to interest contexts, i.e. they may subscribe to interest contexts of a base class, but not necessarily to the corresponding inherited contexts of the subclasses. Finally we can implement abstract interest contexts. An abstract context is a context which is defined for an abstract class. Also the specification of the relations or the events may be abstract as well.

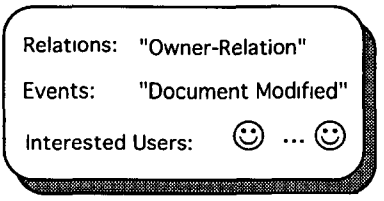

Figure 4 An interest context

4.3. Event Distribution

What happens, if an event gets raised by an object? First, the system checks, if there are matching interest contexts defined for the corresponding object class, i.e that have the newly created event type listed in their event description For all these contexts the system extracts the relation types and forwards the new event to all objects in the original object's surroundings, which are interconnected by one of the relations in this list.

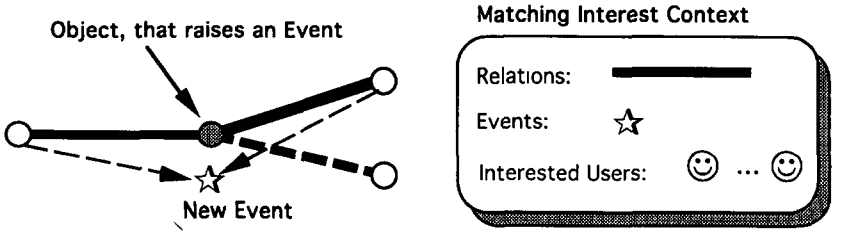

Figure 6 Event distribution

Distribution of events always means an accompanying passing of a reference to the interest context, that led to the event distribution along this specific relation.

This is necessary, in order to determine later on, which user wants to be informed about the event, if he is accessing this object. Furthermore the event object keeps a list of interested users as well. This list is formed by the union of all users that have subscribed to one of the interest contexts involved in the event distribution.

4.4. Event Notification

The distribution of the event according to the interest contexts leads to the presence of this event in a whole space of objects. Furthermore, there are different users that have expressed their interest in the event, and this space of objects is structured according to overlapping subspaces, for each of these users If a user enters such a situation, i.e. if he is accessing one of the objects that take part in such a situation, the system performs a notification about all events that have occurred, since last access*

The notification can be done in different ways and is independent of the core event model. We propose to have different urgency levels for subscription of interest contexts, which determine the form of presentation of event information at the user interface. A high urgency would typically lead to a disruptive notification, such as popping up a message window, whereas a low urgency could reflect the information by a change of color of the object's icon and leave the details of information to explicit user request. After the notification has been performed, the user is canceled from the list of interested users and will not be notified about the event again.

4.5. An Example

Consider the class "circulation folder" which is derived from class "folder" A circulation folder defines a list of recipients which sequentially receive the folder in their private workspace. The class defines the following relations that can be established to the actors in the list of recipients as shown in Figure 5:

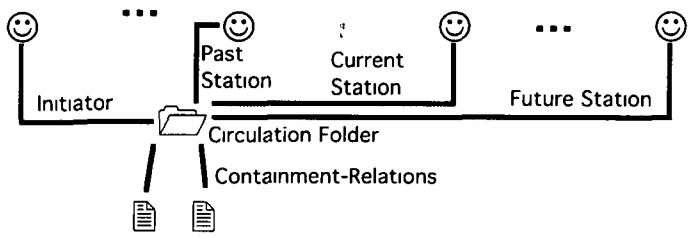

Figure 5 A circulation folder

* This can be determined by an inspection of the corresponing interest contexts for each event, i e by checking, if this actor has subscribed to one of the contexts and by checking, if the user hasn't been informed before (via access of some other object)

- "Initiator" is a relation, which connects the originator of the work flow.
- "Past Station" connects all recipients, that have already finished their task.
- "Current Station" defines the actor, which is currently working on the circulation folder.
- The "FutureStation"-Relation identifies all the users, that will eventually receive the circulation folder.

Additionally a circulation folder inherits the Containment-Relation from it's parent class Each time the folder travels from one station to the next, it raises a "Change Station"-event and the "Current Station"-relation of the former current recipient is exchanged by a "Past Station"-relation. The "Future Station"-relation of the successor station is exchanged by a new "Current Station"-relation.

The class "circulation folder" might define the following interest contexts:

- A "Progress"-context, which uses the "Initiator-relation to describe the situation. If this context is subscribed, users get an awareness about the state of any circulation folder, they have sent away. The class of events could e.g. be the "Change Station" events, such that they are informed, every time the folder changes from one station to the next.
- As a circulation folder inherits all interest contexts from his parent class, the user of the current station can make use of all awareness facilities he has subscribed for the class folder, e.g. he could subscribe to interest contexts that provide an awareness about the work, that has been performed by his predecessors, or he could be informed about things that happen synchronously, if he has opened the folder.
- We can additionally achieve awareness about work to be expected in the near future, with the following "Future Work"-context: the relations of this context are the "Future Station"-relations and as the interesting events we can simply define the "Creation"-event. A user subscribing to this context gets informed about the creation of each circulation folder, where he is contained in the list of recipients.

Interest contexts have to be defined for an object class only once and can be subscribed by any user in the system, who wants to share the corresponding awareness facilities.

5. GroupDesk

In the remainder of this paper we describe the GroupDesk system, a prototype CSCW application, that was specifically developed to demonstrate the event model, presented so far. The design of the system has dropped any features, that would have complicated the investigation of the event related concepts. As a result, GroupDesk has developed as a small platform, supporting distributed work in a simple environment for document production. The second design rationale behind the system has been the evaluation of novel object oriented development

paradigms. For the implementation, a distributed development platform, compliant to OMG's CORBA standard, has been chosen (Object Management Group 1991; Object Management Group 1992).

5.1. GroupDesk Functionality

The system implements an environment for collaborative development and sharing of documents. The basic metaphor for coordinating and structuring cooperative work, used in the system, is the shared workspace. A workspace may be assigned the work artifacts and a set of members, which forms the group of users that have access to these objects and may freely modify them. Workspaces may be thought of as rooms in which the objects are visible and accessible and where the group members see each other and meet in order to perform shared tasks. In addition to the group workspaces, the system establishes a private workspace for each user that is registered in the system. Private workspaces may only be accessed by their respective owners.

Workspaces in GroupDesk allow members non sequential, unrestricted access of the objects they contain, thus supporting the accomplishment of tasks, that require continuous access of documents by the group members. The actual physical location of the artifacts in the distributed environment remains hidden from the users. In order to keep the design of the system as easy as possible, GroupDesk imposes no restriction or semantic prescription on the action of users. There are no conflict avoidance mechanisms implemented, e.g. to prevent two users from simultaneously modifying objects. The system addresses these problems by providing an implicit overview about all activities that are currently going on in the environment and thus enables an awareness of the users to prevent these situations.

The interface of the system presents workspaces as windows. The objects in the workspaces are shown as icons. The members of the workspace are also shown as labeled icons, showing the picture of the corresponding users. Interaction with the system is implemented by the usual drag and drop mechanism: objects may be moved freely around in the workspace and may be arranged as the users prefer.

Interaction with the system may be performed by double clicking on the respective object icon. If the object is a document, the system will launch the corresponding editing tool. Double clicking on folders and workspace icons opens a window, showing the contents of these objects. The system additionally supports synchronous and asynchronous communication facilities, which are attached to the actor icons. Double clicking on these symbols launches a video conference to the corresponding user. Artifacts may be moved into another location, i.e. workspace or folder, by simply dragging the object onto the destination's icon or window and dropping it. Each icon has additionally an associated menu attached, which gives users the possibility to delete, copy or rename the object.

5.2. Architecture

GroupDesk is designed as a distributed CSCW application, consisting of an object server and an arbitrary number of client applications, that may request services. The server manages a repository of objects and is responsible for administration and admission of the users, entering the system. The functionality of the system and the distribution of events and object changes is completely controlled by the server. Furthermore, the server is an instance that keeps the object repository consistent and enables a common view on the overall state of work.

The implementation is based on a CORBA compliant distribution platform which hides the aspects of localizing objects in the domain and granting access to remote objects from the clients. Clients may request services from any object in the system directly and don't have to be concerned with the interaction with the server. Interoperability between different domains is possible, although not yet based on the interaction of different domain servers. Currently users may start a client locally and access a server over the Internet. No matter where the server is running, communication with the server is completely hidden from the user.

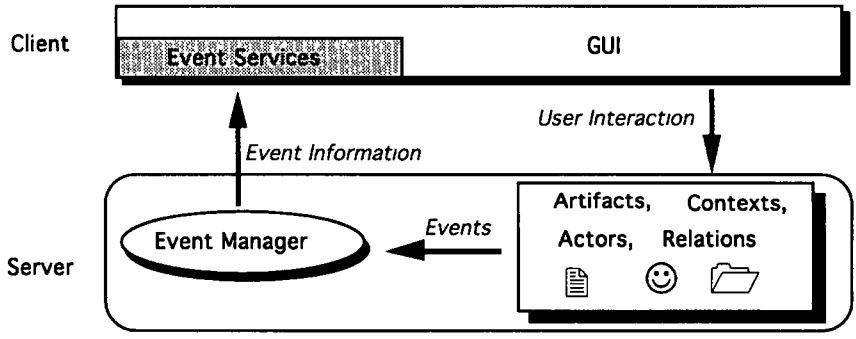

Figure 7 GroupDesk architecture

The system is structured according to Figure 7 On the client side, GroupDesk offers the services, that enables users to interact with the system. This basically consists of the graphical user interface, which is responsible for offering the functionality to access and manipulate objects. Additionally, the user interface displays the changes in the state of objects as well as the dynamics in the work setting, whenever the server notifies it about new events. The client side also provides the management facilities, that allow the user to explicitly request event related information via a history service.

On the server side, GroupDesk implements the common facilities to serve client requests for accessing objects, such as opening documents, deleting objects, or moving entities to another location in the repository. The object repository maintains the representation of the organizational context, i.e. the structuring of

artifacts by different typed relations, to form a semantic network. At the current stage, the system supports structural and operational relation types. The server also implements an event manager, which handles the generation of events each time a user performs some action that results in a change of the object repository and subsequently performs the propagation of the events. The event manager further is responsible for storing events in object related event lists and notifies all interested clients about the changes, that took place. Additionally, it may receive event retrieval requests from clients and access and return event information.

5.3. Awareness Facilities in GroupDesk

The emphasis in developing the GroupDesk system has been the support of user awareness, by applying the strategy of local event distribution, that has been described previously. The event related services present the users the dynamics in the work process. Events caused by other actors and external influences are displayed by the system in an unobtrusive manner and include active notifications of changes in the work setting, as well as inactive presentation of event information on user request.

5.3.1. Events in GroupDesk

Currently GroupDesk has implemented two kinds of activities: presence in a workspace and generic working activities.

Whenever a user enters a workspace, the system adds an operational relation between the actor object and the workspace object. Furthermore an activity event is generated, which describes this action. The event contains a time stamp and a reference to the actor, who has entered the workspace. Subsequently all events, that have happened in the workspace since this user has accessed it the last time, are forwarded to the actor object and the user can immediately see what has changed. The system forwards modification events, that have happened in the past and the currently ongoing activities of other users in the workspace to the new user. Users may also request information about activities that are already finished. This helps to keep the amount of event information small and concentrate on the current state of work.

The generic work activities include any type of action the user performs on an artifact other than workspaces. Currently this involves editing a document or opening a folder. In both cases the system establishes an according relation between the actor and the corresponding object and presents event information related to the object. Among the types of modifications, GroupDesk has implemented object updates, which are generated whenever the content of a document, folder or workspace changes, creation, deletion and movement of objects. For each modification, the system generates a new event object and stores it in an object specific event list.

5.3.2. Event Propagation

Event distribution is currently statically defined: users cannot specify individual interest contexts. This is due to the fact, that the system is currently in an experimental stage and yet lacks many of the concepts that have been presented before. Similarly, the types of relations currently supported have to be complemented. They consist of structural and operational relations. The structural relations support the basic types of relations to structure the work artifacts in workspaces.

A typical GroupDesk scenario is shown in Figure 8. In this example, two workspaces are modeled. Structural relations place objects into the respective workspace context and are also used, to describe the contents of folders. Operational relations consist of two types, that describe presence of actors in a workspace and activities concerning artifacts, e.g. editing a document Artifacts may be shared among workspaces. In the example, a document object is contained in two workspaces simultaneously.

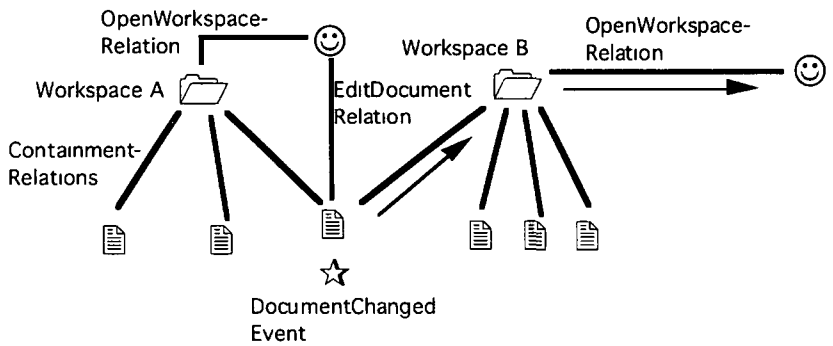

Figure 8 A GroupDesk scenario

In the example the user in workspace A edits a document, which in turn raises a "Document Changed"-event. This event is forwarded along the Containment-relations to all surrounding workspaces, such that the user in workspace B gets a peripheral awareness about the change.

To demonstrate the event model GroupDesk defines a global strategy for event distribution, which cannot be tailored by subscription of individual interest contexts. The distribution of events is defined as follows:

- Structural relations always forward events from the inferior object to the superior object, but not vice versa.
- Operational relations always distribute events to the involved actor.

5.3.3. Event Visualization

The display of event information is integrated in the standard user interface. Modifications on artifacts are indicated by changing the color of the object's icon

Different colors are provided for the different types of modifications. The system however presents only the most recent modification on an object. It has turned out, that this is usually sufficient to give an overview at a glance about the state of affairs in the workspace. If more detailed information is needed, the user may request the complete summary of changes and activities concerning an object via the history service.

Synchronous events, i.e. currently ongoing activities of other users in the same workspace are shown on the graphical user interface by colored connection lines linking the icon of the actor, who is currently performing the activity, with the icon of the object, that is involved in the activity. The icons of workspace members are always shown in the workspace window, even if they are not currently active. If a member enters a workspace, this activity is shown by changing the member's icon from gray scale to colored. Non-members entering the workspace, are indicated by adding their actor icon in the workspace windows of all other users that have opened this workspace.

In general, the visibility of events is restricted to the visibility of objects in the user's view. This means that events are usually shown at the topmost object in the structural hierarchy, which is visible to the user. If the user wishes to see more details, he can open this object and inspect it's contents. As an example consider the modification of an object contained in a folder. If the folder is closed, the user may only notice that some modification happened to the folder. If he wants to see more details, he may open it and inspect it's contents.

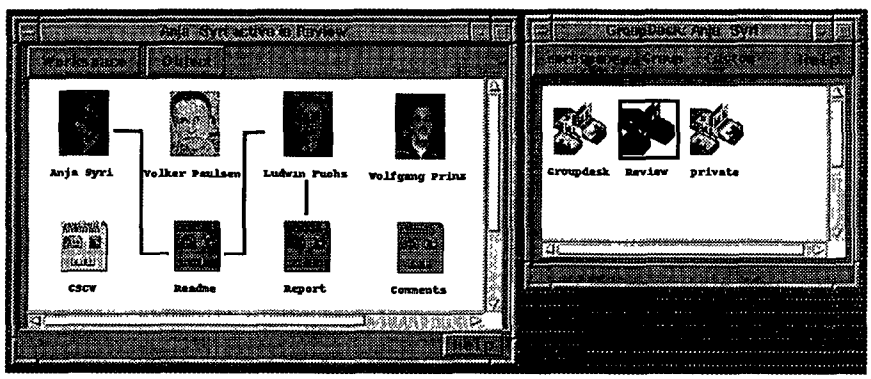

Figure 9 The GroupDesk Interface

A history service allows users to get a detailed description of the events that happened during an object's lifetime. The service is available for any type of object except actors. In the current version the description is text based, but it is planned, to implement graphical display of event information (e.g. charts, showing the appearance of events over time) as well as event filtering and interpretation. It has turned out however, that the current implementation is already quite

useful when it comes to exactly determining what has happened in the past with an object.

Figure 9 shows a GroupDesk Interface. The right window displays a list of workspaces. Workspaces with ongoing activities or changes are displayed in another color. The left window shows an open workspace, with two active users and synchronous editing activities indicated by the connection lines. The modifications on the document are also indicated by different colors of the icons.

6. Future Work

The implementation of the GroupDesk prototype is currently only realizing a minimal environment for experimenting with the concepts of event propagation and support for shared awareness, presented in this paper. In order to capture the whole facilities of the model, many things remain to be done. Most notably, the representational issues need to be extended, i.e. the types of relations need to be extended by semantic relations. Additionally, the existing relation types have to be further diversified. To capture the dynamics, it is necessary to implement the concept of interest contexts for individual tailoring of event propagation.

Conversely the system needs to be enriched with more sophisticated event services on the client side. It is planned to extend the history service with facilities for graphical display of history information. To achieve support for uncoupled awareness, an event notification service has to be integrated. Last but not least, it would be necessary to provide full object persistency in order to make the prototype really usable for practical work. This is currently not realized to a full extent.

It has turned out however, that the approach of presenting the default event information graphically at the user interface results in an implicit overview for the participants in the work process about the state of affairs, without overloading them with too many details.

The concepts presented in this paper will be implemented in the German research project POLITeam (Hoschka, Kreifelts et al. 1994) on the basis of the CSCW platform LinkWorks.

7. Conclusion

In this paper we have presented an event mechanism which is capable of providing information to describe the dynamics and state of work in CSCW applications and thus may be applied to support shared awareness in systems for the coordination of cooperative work The model proposes the representation of the environment as a semantic network. Awareness about changes and synchronous activities in the system is supported by the generation and distribution of events in the semantic network. The propagation mechanism provides the flexibility, to distribute

the information, such that it may be accessed in places, where it is relevant, and on the other hand prevents overloading the user with unnecessary details.

GroupDesk, a prototype implementation of this model has been presented. The system is implemented on the basis of a distributed object service platform. The system implements a simple environment for coordination of distributed work and enables the support of shared awareness for the users by applying the event model and visualizing the event information using the desktop metaphor.

8. References

Belkathir, N and J Estublier (1987). *Software Management Constraints and Action Triggering in the ADELE Program Database*. First European Conference on Software Engineering, Strasbourg, France.

Benford, S. and L E Fahlén (1993). *A Spatial Model of Interaction in Large Virtual Environments*. Third European Conference on Computer Supported Cooperative Work, Milan, Italy

Dittrich, K W , et al. (1986) *DAMOKLES - A Database System for Software Engineering Environments* International Workshop on Advanced Programming Environments, Trondheim, Norway

Dourish, P and V Bellotti (1992). *Awareness and Coordination in Shared Workspaces* CSCW '92 - Sharing Perspectives, Toronto, Canada. ACM Press.

Fuchs, L , U. Pankoke-Babatz, et al (1994). *Ereignismechanismen zur Unterstutzung der Orientierung in Kooperationsprozessen*. German Conference on Computer Supported Cooperative Work, Marburg, Unknown Publishers

Hoschka, P , T Kreifelts, et al. (1994) *Gruppenkoordination und Vorgangssteuerung* Dritter GI Workshop des GI-AK 5 5.1 Betrieblicher Einsatz von CSCW Systemen, Sankt Augustin GMD, GMD Studien

Kaiser, G E and D E. Perry (1987). *Workspaces and Experimental Databases Automated Support for Software Maintenance and Evolution* Conference on Software Maintenance

Object Managemennt Group (1991) The Common Object Request Broker Architecture and Specification OMG

Object Management Group (1992) Object Management Architecture Guide. OMG Tech Document.

Pankoke-Babatz, U (1994). *Reflections on Concepts of Space and Time in CSCW*. ECCE 7 Seventh European Conference on Cognitive Ergonomics. Human Computer Interactions· From Individuals to Groups, Sankt Augustin, Germany, GMD Studien.

Prinz, W. (1993). *TOSCA. Providing Organisational Information to CSCW Applications*. Third European Conterence on Computer Supportes Cooperative Work - ECSCW'93, Milan, Italy, Kluwer

Sohlenkamp, M. and G Chwelos (1994) *Integrating Communication, Cooperation and Awareness. The DIVA Virtual Office Environment* Proc Conference on Computer Supported Cooperative Work - CSCW 94, Chapel Hill, NC

Streitz, N., et al (1992). *SEPIA: a cooperative Hypermedia Authoring Environment* ACM Conference on Hypertext, Milan, Italy