S. Bødker, M. Kyng, and K. Schmidt (eds.). *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work, 12-16 September 1999, Copenhagen, Denmark*

AREA: A Cross-Application Notification Service for Groupware

Ludwin Fuchs
Boeing Mathematics & Computing Technology, USA
email: ludwin.fuchs@boeing.com

Abstract: This paper presents AREA, an integrated synchronous and asynchronous notification service for awareness information. AREA uses a semantic model of the client applications to support cross-application awareness. The service is based on the dichotomy of interest and privacy Notifications of user activities are a function of relevance in the current work situation and the privacy requirements of the involved users The paper motivates the AREA framework and discusses the system in terms of its formal modeling capabilities and its operational aspects This is followed by practical implementation considerations addressing the role of AREA as a groupware infrastructure Finally, a prototype groupware application is presented, which uses the AREA service to support user awareness

Introduction

The Nineties have seen an important paradigm shift in the usage of computer technology. Systems are no longer exclusively seen as number crunchers helping us to solve difficult problems. Instead, driven by the enormous growth in network bandwidth and the ubiquity of the Internet, the computer evolves more and more into an inhabited information space, in which we interact with other people.

One of the most obvious manifestations of this is denoted by the term *awareness* the ability of the technology to expose the activities of the people in the electronic space. In collaborative environments the role of awareness for

coordinating work activities has been shown in numerous studies (see e.g. Heath and Luff, 1992, Rogers, 1993). But support for awareness is not just important in the work environment An obvious proof to this is the commercial thrust, which has been initiated more recently by the need for awareness on the Internet An example is ICQ, an Internet buddy list application, which has led the 1998 shareware download surveys for months with more than 10 Million estimated users online.

This paper focuses on collaborative environments as constituted by a shared information space and a variety of independent tools being used individually as well as jointly among the group members The paper proposes treating awareness as a global, i.e. application independent requirement in such an environment

Based on these considerations the paper presents AREA, an integrated synchronous and asynchronous notification service for awareness information AREA supports cross-application awareness using a model of the clients' semantics to capture the awareness-related operational aspects of the applications. The model explicitly addresses the potential for conflicts of awareness and privacy. Notifications of user activities are a function of relevance in the current work situation and the privacy requirements of the involved users

The emphasis of the paper is to motivate the AREA framework and to discuss the system in terms of its formal modeling capabilities and its operational aspects. This is followed by practical implementation considerations about the role of AREA as a groupware infrastructure Finally, a prototype groupware application is presented, which uses the AREA service to support user awareness.

Design Issues

Supporting group awareness creates difficult design problems for developers of distributed collaborative applications, problems with both technical and organizational dimensions..

Technically, the developer is faced with the heterogeneity of applications. Groupware can only be commonly characterized by the weak incentive to support collaboration between a number of users. Workflow systems, shared design tools, email, collaborative web applications, and document management systems all are first class groupware systems. These technologies lack a common base that could be abstracted out to form the core of a formal model of groupware and eventually inform the design of awareness support. Supporting awareness in a shared text editor is a fundamentally different problem than supporting awareness in a persistent discussion space such as Lotus Domino.

A further challenge is introduced by the global nature of the awareness problem collaborative environments consist of a multitude of tools and applications. Performing a collaborative task rarely is a continuous and concerted activity and almost never maps to a single technology that could support it.

Instead, working together is opportunistic: users frequently switch between a number of tools running in parallel and also the shared information crosses the boundaries of applications. Collaborating requires working in synchronous and asynchronous mode, frequently switching between modes, or even working in both modes at the same time. We talk on the phone while we browse our email or look up information in a shared discussion database. Writing a shared document involves working at different times on individual parts followed by meetings to discuss and merge them together. Supporting awareness thus becomes a global issue. What we really need is an integrated view on the activities in the work environment as a whole, covering many applications, rather than just one. Even if we have a good formal model of a particular groupware application domain, the provision of awareness will be restricted to the boundaries of a single tool and lack the global perspective.

Global awareness support is not only difficult to achieve architecturally, it also introduces new problems. If information about user activities is collected and disseminated in the whole work environment, it is likely to overload the user with irrelevant and out-of-context notifications. To prevent this, users need a way to define their information needs in terms of the work they are performing The same event can be highly important for one user and completely uninteresting for another user in the same situation. Similarly, an event may be important for a user in some situation and irrelevant for the same user in another

Supporting individual awareness leads to a number of socio-technical conflicts These result from two constellations of conflicting goals between the individual user and other users or the group as a whole: (1) the user's demand for privacy creates a conflict with other users' or the group's demand for awareness and (2) the user's goal to reduce information overload clashes with other users' or the group's goal to establish common reference.

The first issue also has organizational and legal dimensions. In many countries data protection laws restrict the organizational usage of personal data such as awareness information. The second issue is particularly important in the group context Only if awareness of activities provides a common reference to all members of the group can it be a vehicle for coordination in a team environment For example, a design team working on a model of a particular section of an airplane often faces many interdependencies between the individual sub-parts. Changing one part can violate constraints imposed by another part maintained by other group members. When a designer changes a part of his model it is critical that these changes are reliably visible to the other group members in order to minimize disruption caused by conflicting changes.

If users can individually tailor awareness notifications, others can no longer be certain that their actions are perceived by the group as a whole, i.e. the activities no longer form a common reference in the group. Technology intended to support awareness needs to address this fundamental tradeoff.

Awareness Frameworks

The inherent complexity of supporting awareness as outlined in the last chapter has fueled the development of awareness frameworks that provide a simplified model of collaboration for particular application domains. They offer basic mechanisms that can be used by systems designers in order to make the actions in the environment visible to others. The following section provides a brief review of various awareness frameworks.

A number of approaches address synchronous collaboration. Gutwin et al. (1996) describe a shared editing environment providing awareness based on multi-user GUI mechanisms. The system offers a number of awareness widgets showing the current focus of others in the application using distortion techniques Activities close to the user's own focus are visible whereas those parts of the document with no ongoing activities are only visible in a global minimized layout view. Thus the authors show that an overview in its physical sense can support user awareness The design of such multi-user interface components can be largely supported by toolkits such as GroupKit (Roseman and Greenberg, 1992) or Habanero (Chabert et al., 1998). More recently, Smith and O'Brian have proposed a similar GUI-oriented approach for awareness in 3D virtual environments (Smith and O'Brian, 1998).

While the user interface approach provides useful mechanisms for developing synchronous relaxed WYSIWIS applications, it doesn't address awareness as a groupware infrastructure issue. This is one of the goals of the Notification Service Transport Protocol (NSTP; see Patterson et al., 1996), an application independent notification protocol for synchronous groupware. NSTP is based on a client-server model, where the server maintains a number of places managing the shared state. Clients can enter any number of places. They can manipulate the shared state and receive automatic notifications whenever state changes occur in the places they have entered. NSTP is a lean framework strictly focusing on efficiency. Consequently, it is free of any assumptions concerning the application's semantics Another synchronous application-independent notification service similar to NSTP is Corona (Shim et al., 1997) Unlike NSTP, Corona provides a persistent data store for the shared state and adds more flexibility for updating the shared state.

All awareness architectures discussed so far focus on synchronous groupware Among the asynchronous models, GroupDesk, a shared workspace prototype, implements awareness using relationships between artifacts in the work environment to notify users about activities (Fuchs et al., 1995) The asynchronous behavior in the GroupDesk awareness model results from distributing events between the artifacts and storing them persistently. Notifications are triggered as soon as interested users access the artifacts The usage of object relations has been adapted in the Aether model to support

asynchronous awareness in 3D populated information spaces (see Sandor et al., 1997). A further asynchronous awareness model is implemented in the Interlocus prototype (Nomura et al., 1998). Interlocus monitors individual user activities and creates snapshots of the objects when the user makes changes. Awareness is provided by intelligently merging individual snapshots and providing chronological views on the documents in the workspace.

The AREA Awareness Model

In the following sections the AREA awareness model is introduced Similar to NSTP, AREA is an attempt to provide an awareness infrastructure component for collaborative environments In contrast to NSTP, AREA is not only an application development environment but may also be used by preexisting groupware tools. Similar to GroupDesk, AREA implements situation-oriented awareness based on relevance relationships among the shared objects. Unlike GroupDesk, AREA offers an open service interface enabling multiple applications to specify event distribution, user-defined interest and privacy specifications, as well as awareness-related group facilitation mechanisms.

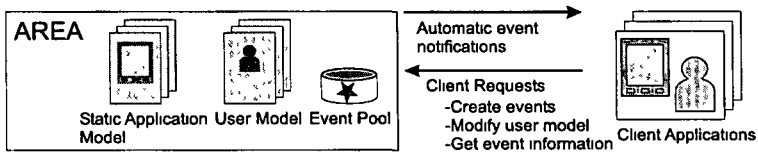

Figure 1: Client-server communication in AREA

AREA applies a client-server architecture, where the server manages the shared state and clients receive automatic event notifications. The shared state consists of a static description of the application semantics, a dynamic user model[1], and a persistent event pool. Client applications can issue requests to create events or to access the user model. Create-event requests cause the server to add a new event to the event pool and issue notifications to other clients.

AREA uses the semantic model of the applications and the user models to determine which clients receive an event notification. This decision is based on the principle of mutual exclusion of awareness and privacy in the following way: a user *A* can be aware about the actions of another user *B* if *A*'s user model specifies interest in the action and there are no conflicting privacy requirements of *B* that prevent the notification.

[1] Note that the notion of a user model in AREA does not mean the system is making inferences based on user actions at the application's GUI, as it is usually the case in adaptive systems The model is solely updated based on explicit requests issued by the application

Approach

AREA supports awareness by providing notifications about *activities* performed by users in any of the client applications. AREA captures activities in terms of *events*. As users perform activities in the collaborative environment, the system creates new events describing these activities For each new event AREA determines all users who will be notified about the event by evaluating all user profiles according to their interest specifications and matching them against the privacy requirements of the performing actor This is illustrated in Figure 2:

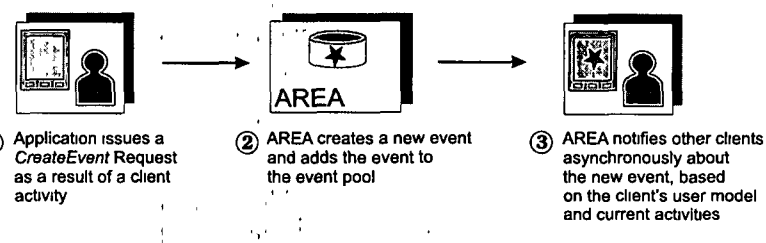

Figure 2 Event notification in AREA

The notions of interest and privacy are at the core of the AREA awareness framework Both terms can be defined very specifically or may be kept very general. An actor's interest can cover a broad range of situations common to the kind of work she performs. It can specify a higher degree of importance for more special situations or it may completely exclude events for some situations.

The Application Model

The awareness framework in AREA is based on four basic model components: actors, relations, events, and artifacts The following paragraphs describe each of these components and discuss how they can be used to describe the application semantics

Actors

Actors denote users of the client applications. An actor is usually a real person working in the environment but any actively performing entity can be registered as an actor in AREA, e.g a software agent. Actors are equipped with a profile describing their interest in the actions of others as well as their privacy requirements.

Artifacts

Each action in AREA involves an actor and an artifact, which is the object of the action In document management systems artifacts typically consist of documents, discussion items, or folder hierarchies. In a technical design application for

building airplanes they might consist of drawings, geometric objects, or more abstract entities like constraints that need to be met Artifacts may have a long lasting life cycle (e g a jointly produced research paper) or they may exist only for a brief time period, such as a telepointer in a multi-user brainstorming tool.

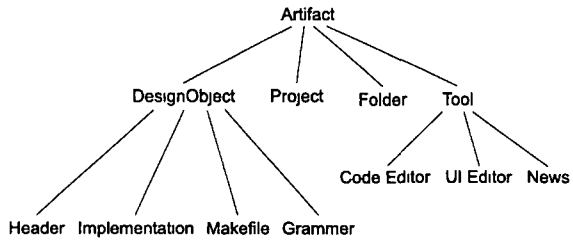

Figure 3: A simple artifact class hierarchy for a software development application

Artifacts belong to classes The system defines a class hierarchy, which is shared among all applications using the system. Figure 3 gives an example of a class hierarchy for artifacts that might appear in a software design environment. Note that the primary criterion of introducing a class in the hierarchy is not sharing but rather whether the usage of the artifact is worth being aware of. Thus, although a news reader rarely is used collaboratively it can be interesting for group members to be aware that another is reading news and hence shouldn't be disturbed.

Applications may extend the class hierarchy with their own specific classes or they may reuse existing classes. For example, a shared drawing application could add its own artifact classes for application specific items, such as the various drawing tools and filters but it can reuse the classes for JPEG- and GIF-images, already defined by another application.

In this way, the class hierarchy maintained in AREA forms a unified set of classes for all artifacts in the work environment that are suitable candidates to be aware of. Note that AREA does not actually maintain instances of the artifact classes The class hierarchy in AREA is only an abstraction for the application entities for the purpose of describing awareness behavior. AREA only maintains artifact references (e g as attributes of events). The actual instances only exist in the applications and will almost certainly belong to a completely different application-internal class hierarchy. The application may not even be object-oriented at all

Events

Events formally describe actions performed on an artifact. Each event has at least 3 attributes: the actor responsible for the event, the corresponding artifact reference, and a time stamp Similar to the artifacts, events are structured in a class hierarchy that defines the semantics of the events. An application can define

new event classes. A new class inherits the attributes of its parent class and may add additional attributes.

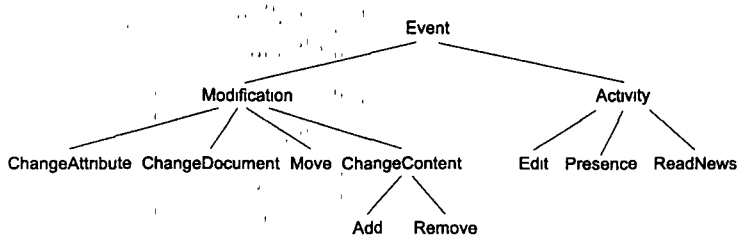

Figure 4 An example for an event class hierarchy

Figure 4 shows an example of a suitable class hierarchy for events, which users might want to be aware of in the design environment example. Note that not all events can occur on all artifacts. For example, *ChangeContent* events can only be created by compound objects like folders. AREA keeps a mapping between event classes and the artifact classes, which can create these events.

Formally, events in AREA denote state changes and thus do not have a temporal duration But the system defines a specific class of events that can be used to describe actions covering a continuous time span. These are called *activity events*. Activity events consist of a *begin-event* and a matching *end-event* marking the time frame of the user activity. In AREA the term "being active" implies that the system has created a begin-event but not yet received a request for an end-event. AREA tracks begin-events in an activity list and uses this information to determine if actors should be notified synchronously.

Relations

AREA uses relations to describe collaborative interdependencies in the application domain Relations are always *1:n* and can only exist between artifacts In contrast to artifacts and events, AREA does not define an extensible class hierarchy for relations. An application can only define relation instances, which may be assigned to one or more of the following predefined groups of relations: *components*, *compositions*, and *situations*.

A component defines a part-of relationship between two artifacts. A composition usually is the inverse relation of a component and yields the aggregate for a given part. A formal requirement in AREA is that for each component there exists at least one composition, which is denoted as the inverse of the component. Note that different component relations can be assigned the same inverse relation.

Situations are relationships between artifacts expressing relevance in terms of collaboration For a given artifact a particular situation defines all other artifacts that share a common awareness-relevant relationship with this object. Thus, the

set of situations for an artifact should be defined such that it makes sense to be aware of events of this artifact whenever the user accesses another artifact belonging to the situation

Relation	If available for an artifact it yields	Is defined for	Relation groups
Identity	the same artifact	all artifacts	Component, Composition, Situation
ContainedIn	all superior folders	all artifacts	Composition, Situation
Contains	the content artifacts	folder artifacts	Component
Dependents	the design artifacts with a software dependency	implementations	Situation

Table 1 Some example relations

Table 1 shows an example of 4 relations, which could be used in the software environment example. The *Identity* relation is defined as component, composition, and situation simultaneously (since each artifact is a part as well as a composition of itself). All relations except the *Contains* relation are defined as situations, since each defines a useful awareness situation for the notification of events.

The User Model

Having introduced the four basic components of AREA, we can now turn to the user model and the operational aspects. A user model consists of a list of interest specifications and a list of privacy requirements. Interest specifications have four parts:

- *The event description* specifies the actions the user wants to be aware of. It consists of a predicate over the set of available event classes and attributes.
- *The scope* defines the space of artifacts for which the description is valid. This in turn involves a predicate expression over the artifact class hierarchy and a component relation. The predicate is evaluated against the artifact raising a new event. The role of the component is to enable indirect specifications, e.g. for all artifacts in a container This introduces additional flexibility for defining interest descriptions.
- *The situation* defines when notifications about the events shall be issued to the actor. They only consist of a situation relation.
- *The intensity* consists of a discrete value that will be interpreted by the application to determine how a notification should be performed at the GUI.

The following figure shows an example of two interest descriptions.

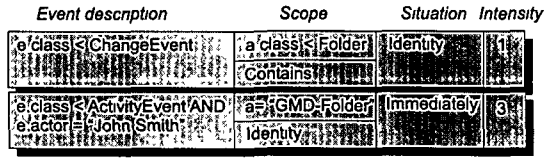

Figure 5 Two example interest descriptions in a user profile

The first description defines interest in all change-events.[2] The scope of the first description has been specified indirectly: it covers the content of any folders A notification of a matching event occurs when the user accesses the artifact creating the event (since the situation is the Identity relation). The value of the notification intensity is 1, i.e. the event notification should be performed in the lowest intensity. The second interest description defines interest in the activities of user John Smith. The scope of this description only covers one particular artifact "GMD-Folder" The situation *Immediately* is a predefined symbolic situation relation. If it is used in an interest description, a notification takes place immediately or as soon as the actor enters the environment

Event Notification

The sum of all interest descriptions in all user models drives the event notification in AREA. Figure 6 illustrates the notification algorithm:

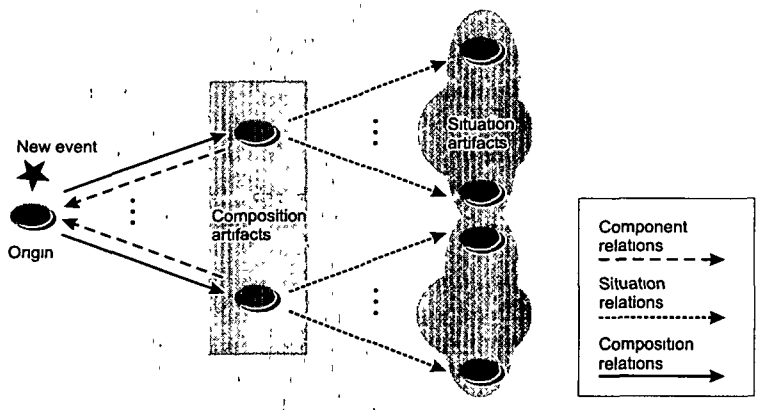

Figure 6 Determining the notification situations for new events

[2] The operator < yields the value *true*, if the left operand is a base class of the right operand *e* and *a* are keywords that are matched against the new event and the artifact creating the event, respectively

When an application sends a create-event command AREA adds a new event instance of the requested class to the event pool and fills in the values of the attributes At the minimum these consist of the reference of the affected artifact (subsequently called origin), the actor, and a time-stamp.

The next step consists of determining the set of matching user profiles, whose scope covers the origin. These are determined by examining all composition artifacts of which the origin is a component A user profile is valid if the following conditions are met: (1) the event description matches the new event, (2) the predicate expression in the scope-component evaluates to true, and (3) the component-relation in the scope-specification is the inverse of the current composition.

With this, each of the composition artifacts to which the origin belongs can be associated with a list of matching interest descriptions These interest descriptions determine the actual situation artifacts that upon access will trigger a notification of the new event. They are determined by applying the situation relations in the remaining interest descriptions on each composition artifact.

The final result of this approach is a list of situation artifacts, each of which is associated with one or more actors to notify and a corresponding maximum intensity value for the notification. At this point a synchronous event notification takes place for all actors who are listed as being active on one of the situation artifacts in AREA's activity list. For all other actors AREA saves the result of the event distribution and notifies them asynchronously as soon as they perform an activity on one of the situation artifacts

Privacy

In order to motivate the privacy strategy of AREA it is important to review the situation-oriented awareness framework presented so far. AREA restricts awareness about the actions performed on a particular artifact to the set of awareness situations available for the artifact's class in the static application model. At runtime such a situation applies if an actor accesses one of the artifacts constituting the situation. In a reasonable implementation, situations will most often comply with the formal organization of work in terms of access and group collaboration.

As an example, consider an application such as a shared drawing tool in which documents can be organized in projects. a useful user model could specify notifications about changes of drawings when the user is active in a project containing the drawing. The situation "when active in a project containing the drawing" implies that the user needs to have access to the project in order to see what happens with the drawing contained therein and is thus compliant with the work organization.

However, AREA also allows the definition of situations that aren't compliant

to the information access restrictions in the work setting. As an example consider a document management application Here it could be useful to have a *RelatedDocument* situation· when accessing a particular document this situation notifies about events of other documents sharing some properties in terms of content. These documents would not necessarily have to be located in the same place. In fact, they could exist completely outside of the document space accessible to the user

Thus a reasonable strategy to enforce privacy in AREA can be obtained by restricting the set of users that may receive events for a given situation Since it is very common that users are willing to expose their activities to others, if they have access to the corresponding work artifacts, this strategy requires users only to deal with notification situations that cross the border of shared access.

Formally, a privacy specification in AREA looks similar to an interest description: It consists of an event description, a scope, and a situation, with each of these components having analogous meaning. The event description defines for which actions a user wants to define privacy, the scope defines the artifact space on which the· specification applies, and the situation-relation denotes the particular awareness situation that shall be restricted for other actors Instead of an intensity value, the fourth component of privacy specifications defines an admission list of actors, listing those that are granted the right to receive the specified event.

The event notification accounts for privacy specifications in the following way: For each new event AREA determines the set of privacy descriptions defined for the responsible actor in the scope of the origin. Formally, this is similar to the first step in the evaluation of the interest profiles. For each composition artifact there is now a list of matching interest descriptions as well as a list of matching privacy descriptions. These privacy descriptions can then be grouped according to the situation they apply to. If no privacy descriptions exist for a particular situation, all actors receive the event notification according to their interest specification. If a privacy description does exist for a particular situation, only those actors listed on the admission list can receive notifications.

Global User Models

AREA does not directly include groups as components of the awareness framework. Nevertheless, the model does support a group-oriented notion of awareness: A set of global user models can be used to enforce the existence of interest and privacy specifications for individual actors. Formally, global user models have the same characteristics as the standard models, except that they apply for each actor in the environment.

Using global user models an application can enforce the visibility of events for each actor and hence create a common reference for those actions. In contrast,

global privacy descriptions make sure that nobody in the system can receive notifications about the specified events in the corresponding situation. In this way, legal and organization-wide regulations in terms of privacy can be enforced. Note that global user profiles cannot be used to prevent an actor from enhancing his privacy.

How does this help to establish group policies? The key to achieve this is to make use of the flexible scoping of the global user models Coordination of group activities in collaborative environments always involves managing common artifacts By defining global user models scoped to these shared artifacts it is possible to establish group policies for awareness notifications.

AREA as an Awareness Infrastructure

So far, AREA is described as a formal framework for asynchronously managing event notifications in a collaborative object space The framework uses a semantic model of the application to allow event notifications in application-relevant work situations. While many aspects of the application model can be defined statically, others require close collaboration of the application and AREA at runtime. The following sections concentrate on the role of AREA as a groupware infrastructure component.

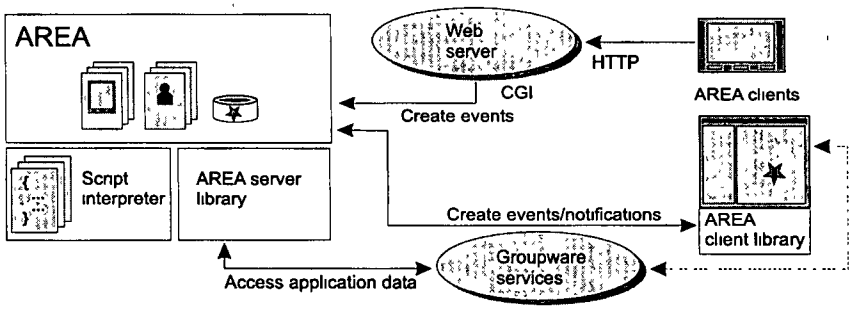

Figure 7 Architecture alternatives for clients using AREA

Figure 7 shows various options for client-server communication using AREA. On the client side there are two alternatives to access the service: using a client library or using a HTTP/CGI interface. Only clients using the library are capable of receiving event notifications. Obviously, this is only possible for newly developed applications, where the source code is available, or in applications capable of extending their functionality using dynamic link libraries. Besides creating events and receiving event notifications, the client library provides access to the details of the static application model and the user models as well as a query interface to the event database. This information needs to be accessed by

the client in order to interpret event notifications triggered by the server.

Clients using the HTTP interface can only create events. Although restricted in functionality this interface offers a perspective to add a range of standard applications as clients of AREA. A prerequisite for using this alternative is the availability of a scripting component, or a macro processor capable of issuing HTTP requests. However, many office productivity tools include such an interface. In this way, AREA can support awareness about activities, such as reading email or working with a word processor.

On the server side AREA provides two alternatives for defining the static application model: using a script interpreter and using a server-side library The server side library contains stubs, which can be used to generate portions of the static application model. A common use of the library is to access the server component of groupware applications or general infrastructure services such as a directory service. In this way, portions of the application model can be generated.

In addition to the server-library, the application model can also be described in a scripting language. The language is particularly useful to define class-level attributes, e.g. event life cycle and persistency properties as well as user-friendly naming for the model components. The dual approach of scripting and using a library offers additional flexibility to define the dynamic portions of the model components: the scripting language offers a construct to call function stubs that are defined using the library, thus providing means to extend the functionality of the scripting component. Since the evaluation of relations and the computation of event attributes are performed on the server side, accessing client side functionality plays a critical role in the event distribution.

User Interface issues

AREA abstracts from the actual application's semantics Consequently, the system doesn't make assumptions concerning the user interface. A notification in AREA means the application receives a description of the new event and the corresponding intensity value. It is completely in the application's responsibility how to display the event at the user interface An application may choose among a variety of different user interface techniques, e g. sound, symbol animations, color annotations, or standard techniques such as dialog boxes, whichever technique fits the application's user interface metaphor and meaning of the intensity.

Cross-Application Awareness

AREA is a cross-application awareness service. To a user working in application *A* the system can provide notifications about activities of another user working in a distinct application *B*. Obviously, this requires the notifying application to be capable of interpreting events in the domain of *B*. This requirement is addressed in AREA by giving clients access to the complete static application model The

class hierarchies for events and artifacts contain user-friendly naming support, such that an application can display meaningful event information about any event

An Example Application

The following sections describe PoliAwaC, an application of AREA in a document management groupware application The application has been implemented in the POLITeam[3] project developing groupware technology for a German federal ministry distributed between Bonn and the new German capital, Berlin (see e g Prinz et al., 1998).

POLIAwaC extends an existing document management system (Digital's LinkWorks) with an Interface providing synchronous and asynchronous awareness. The system uses AREA for the management and notification of events generated by the core document management server and office tools used to manipulate the documents. The following description concentrates on the application model of POLIAwaC in AREA and discusses the resulting support for awareness in terms of the available notification situations. The details of the system and the user interface mechanisms are described by Sohlenkamp (1998) and Mark et al. (1997)

The application model

The following figure provides an overview of the static application model of POLIAwaC as it is defined in AREA:

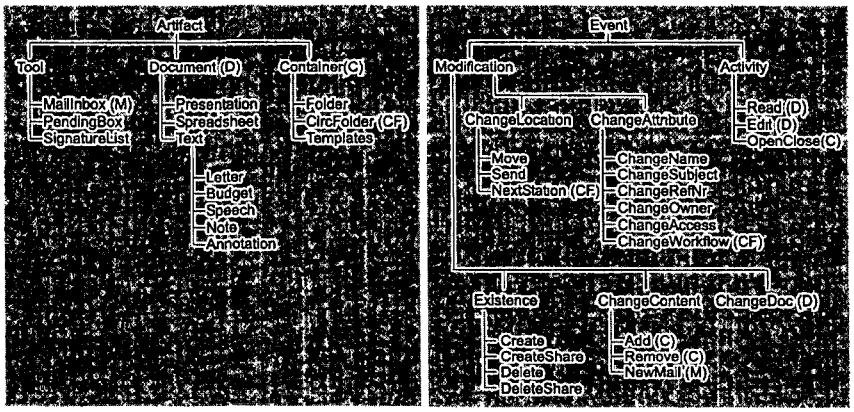

Figure 8 The class hierarchies for artifacts and events in POLIAwaC

[3] POLIAwaC = POLITeam Awareness Client

The left side of Figure 8 shows the artifact class hierarchy Since POLIAwaC is a document-sharing environment for an administration, the most important artifacts are the different types of text documents. The other classes represent the tools and shared container that are available in POLIAwaC The class *CircFolder* denotes a special folder offering some basic document routing capabilities

The right side of Figure 8 illustrates the event class hierarchy POLIAwaC defines three classes of activity events. Opening and closing folders and editing and reading documents. All other events are modification events, i.e events describing discrete state changes. The abbreviations behind class names correspond to some of the artifact classes on the left If present, they indicate that an instance of this event class can only be created by artifacts of the corresponding artifact class. Otherwise, any artifact can use the event class.

The following table lists some of the relations defined in POLIAwaC:

Relation	If available for an artifact it yields	Is defined for	Relation groups
Identity	the same artifact	all artifacts	Component, Composition, Situation
Immediately	N/A (symbolic relation)	all artifacts	Situation
ContainedIn	all superior folders	all artifacts	Composition, Situation
Contains	the content artifacts	folder	Component
SameProcess	all artifacts with the same reference number	all documents	Situation
Workflow	all documents belonging to the same workflow	all documents	Situation

Table 2 Some example relations in POLIAwaC

The table lists the most commonly used relations in POLIAwaC. Similar to the example in Table 1 the system defines a *ContainedIn*-relation, which can be used for indirect interest descriptions. Among the situations, *SameProcess* yields all artifacts with the same reference number[4]. The *SameWorkflow* situation covers all documents in a circulation folder.

The application model allows users to create very specific (e g "notify all ChangeDoc-events of text 'report99' immediately with high intensity") or very general interest descriptions (e.g. "notify all activity events of objects when I am active in the corresponding workspace with low intensity"). POLIAwaC enhances the usability of the model by allowing users to share the functionality of predefined interest descriptions and by using a wizard interface to define new ones. Also, POLIAwaC applies several simplifications in the model to ease the definition process, e g the predicates in the scope specification are simplified to match only single artifact classes or instances.

The facilities for defining privacy have been hidden completely from the user.

[4] Reference numbers are used throughout the application in order to assign documents to a particular process

For each artifact, event notifications are restricted to those users having access to the artifact. Technically, this is achieved by automatically defining privacy descriptions for those situation relations that are not compliant to the formal work organization. This applies only to two relations: *Immediately* and *SameProcess*.[5] This strategy emphasizes the need-to-know principle for the dissemination of awareness information while at the same time reducing the cognitive complexity of interest specifications It also guarantees that the privacy of awareness information is compliant with the degree of information sharing, i.e. private spaces are also private in terms of awareness information.

Notification mechanisms

Figure 9 shows the application's main window. The client uses the standard desktop metaphor. The document hierarchy and the contents of opened containers are displayed in different windows. Users have the possibility to define different views on objects regarding sorting criteria and iconic or textual display, thus allowing for individual working styles.

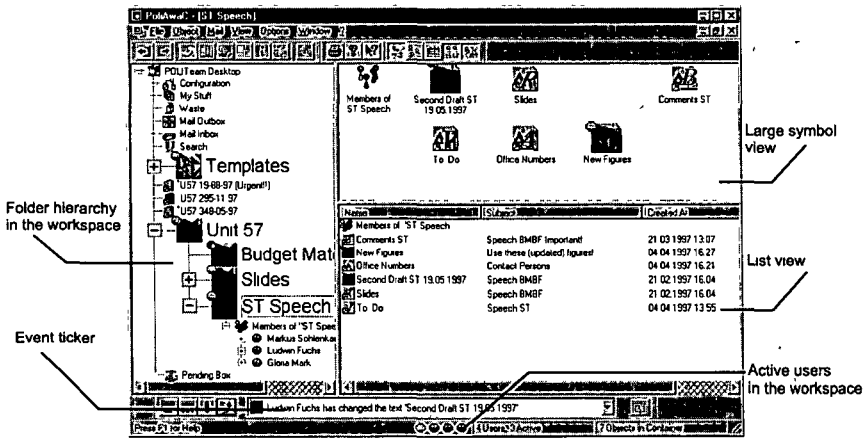

Figure 9: The main application window of POLIAwaC

The notification techniques employed in POLIAwaC depend on the class of event and the event intensity Four different intensity levels are defined in POLIawaC with the respective UI techniques being more disruptive as the intensity increases.

In the lowest intensity, events are displayed using symbol embellishments for activity events and color overlays for change events. In the next higher intensity level, POLIAwaC uses icon enlargements, which are particularly useful to signal

[5] Artifacts can belong to the same process without being accessible in a common folder or workspace

past and ongoing activities in the folder hierarchy shown in the tree view on the left side of the main application window. Events in the third intensity level are displayed as a message in an event ticker at the bottom of the main window. The ticker widget can be used to browse through the list of recent events. Events with the highest intensity value are displayed in a modal dialogue box, which has to be acknowledged by the user. Thus the low intensity levels cause events to be perceivable using peripheral vision while the higher levels force users to focus on the information presentation explicitly.

Conclusions

The experience with POLIAwaC has shown that AREA provides a very powerful tool for adding awareness support to a groupware environment. Unlike alternative notification services such as NSTP, AREA requires an extensive model of the application domain to be supplied. The POLIAwaC example has shown that with a relatively lean domain model the usefulness of awareness notifications can be enhanced substantially. Nevertheless, the creation of this model is a significant overhead in using the service. The following benefits justify these additional costs:

First, having an integrated domain model is a prerequisite for cross-application notification. Applications can only exchange and present events that are occurring in another application in a meaningful way, if they have access to a semantic model of the events.

Second, the application model offers a homogeneous interface to specify work-oriented support for awareness. Events can be notified according to individual and group interest profiles such that notifications occur at the right time, i.e. synchronously, if the user happens to be in the right situation at the time the event occurs, or asynchronously, as soon as the user enters the appropriate notification situation.

Third, the availability of the application model frees client applications from having to deal with the complexity of notifying events in the appropriate situation and enforcing privacy constraints. By maintaining a unified application model AREA provides a single interface to the applications as well as to the user for the delegation of this complexity.

Privacy is considered as an integral requirement for providing awareness in a groupware setting. The notification service takes a 0-1 approach to deal with conflicts resulting from incompatible privacy and interest specifications in the user models. Event notifications only take place, if there is no conflicting privacy profile. This strategy is clear and simple and guarantees that organizational and legal restrictions as well as local group policies can be applied. However, this approach may not be flexible enough for some applications in which a continuous degradation of the awareness notification is desired in response to enhanced

privacy demands.

As an application independent groupware service AREA requires the application to supply the user model AREA only provides an interface for specifying interest and privacy. In the POLIAwaC example a dual approach was taken: the client provides a dedicated user interface to specify interest profiles but privacy profiles are defined automatically, hidden from the user. In general, using AREA requires deciding who is responsible for setting the awareness parameters. The experience with POLIAwaC indicates that it may be desirable to provide more support on the side of the service for this. For example, it could be helpful to have an independent tool to define general group- and work-oriented awareness strategies For example, instead of the need-to-know strategy for privacy taken by POLIAwaC it might be desirable to realize a reciprocity-based privacy strategy. Identifying suitable strategies for privacy and interest and making them available in AREA requires further research.

What are the technical limitations of the AREA awareness model? The system is restricted to notify discrete events rather than provide awareness using continuous media streams. Also, the event distribution mechanism can be computationally demanding, depending on the underlying application model The biggest impact results from the evaluation of relations As a consequence, AREA is not suited to disseminate synchronous, time critical events typically found in synchronous 3D environments. However, situation oriented event notification has the potential of being a useful strategy for synchronous application sharing (e.g to support relaxed WYSIWIS). Determining the borderline of applicability for synchronous shared applications remains to be investigated.

Acknowledgements

The research described in this paper has been performed when I was working at the German National Research Institute for Information Technology (GMD-FIT). I want to thank all the members of the POLITeam project for their great support Also I want to thank Steve Poltrock at Boeing and the reviewers of the ECSCW '99 program committee for their comments.

References

Chabert, A , Grossman, E , Jackson, L , Pietrowicz, S , and Seguin, C (1998) "Java Object-Sharing in Habanero", *Communications of the ACM*, Vol 41, No 6, pp 69-76.

Fuchs, L , Pankoke-Babatz, U , and Prinz, W (1995) "Supporting Cooperative Awareness with Local Event Mechanisms The GroupDesk System", in *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW'95)*, Stockholm, pp 247-262

Gutwin, C , Roseman, M., and Greenberg, S. (1996): "A Usability Study of Awareness Widgets in a Shared Workspace Groupware System", in *Proceedings of the Conference of Computer Supported Cooperative Work (CSCW'96)*, Boston, MA, ACM Press, pp 259-267

Heath, C , and Luff, P (1992) "Collaboration and control crisis management and multimedia technology in London underground control rooms", *Computer Supported Cooperative Work*, Vol 1, No 1-2

Mark, G , Fuchs, L., and Sohlenkamp, M (1997). "Supporting Groupware Conventions through Contextual Awareness", in *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, UK, Kluwer Academic Publishers, pp. 253-268

Nomura, T , Hayashi, K , Hazama, T., and Gudmundson, S. (1998) "Interlocus Workspace Configuration Mechanisms for Activity Awareness", in *Proceedings of the International Conference on Computer Supported Cooperative Work (CSCW '98)*, Seattle, WA, ACM Press, pp 19-28

Patterson, J F , Day, M , and Kucan, J (1996) "Notification Servers for Synchronous Groupware", in *Proceedings of the Conference of Computer Supported Cooperative Work (CSCW'96)*, Boston, MA, ACM Press, pp 122-129

Prinz, W , Mark, G , and Pankoke-Babatz, U (1998) "Designing Groupware for Congruency in Use", in *Proceedings of the International Conference on Computer Supported Cooperative Work (CSCW '98)*, Seattle, WA, ACM Press, pp. 373-382

Rogers, Y. (1993) "Coordinating Computer-Mediated Work", *Computer Supported Cooperative Work*, Vol. 1, pp 295-315

Roseman, M , and Greenberg, S (1992) "GroupKit A Groupware Toolkit for Building Real-Time Conferencing Applications", in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '92)*, Toronto, Canada, ACM Press, pp 43-50

Sandor, O , Bogdan, C., and Bowers, J (1997) "Aether An Awareness Engine for CSCW", in *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, UK, Kluwer Academic Publishers, pp. 221-236

Shim, H S , Hall, R W , Prakash, A., and Jahanian, F (1997) "Providing Flexible Services for Managing Shared State in Collaborative Systems", in *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, UK, Kluwer Academic Publishers, pp. 237-252

Smith, G , and O'Brian, J (1998) "Re-Coupling Tailored User Interfaces", in *Proceedings of the International Conference on Computer Supported Cooperative Work (CSCW '98)*, Seattle, WA, ACM Press, pp. 237-246

Sohlenkamp, M (1998) *Supporting 'Group Awareness in Multi-User Environments through Perceptualization*, dissertation thesis, University Paderborn, Institute of Computer Science, Paderborn, Germany Also available at http //orgwis gmd de/projects/POLITeam/poliawac-/ms-diss