Aether: An Awareness Engine for CSCW

Ovidiu Sandor[1], Cristian Bogdan[1], and John Bowers[1,2]
[1] CID and IPLab, The Royal Institute of Technology, Stockholm, Sweden
[2] Department of Psychology, University of Manchester, Manchester, UK
ovidiu@nada.kth.se, cristi@nada.kth.se, bowers@hera.psy.man.ac.uk

Abstract: Extending and reinterpreting earlier work on the 'Spatial Model', this paper presents a generic model for supporting awareness in cooperative systems ('the AETHER model') and an implementation of a prototype awareness engine. The applicability of the approach is investigated by showing how some of the fundamental functionality in CSCW applications (e.g. versioning and access control) can be supported by the engine and by 'simulating' some other applications from the CSCW literature The paper closes with a discussion of how the model facilitates the construction of flexible CSCW systems (e.g. workflow systems) supporting a variety of forms of awareness.

Introduction

The topic of awareness has received a great deal of attention in recent work in Computer Supported Cooperative Work (CSCW). Providing participants with mutual awareness of each other's activities is often seen as a important research and design goal (e.g. Dourish et al., 1992; Tollmar et al., 1996). The emphasis of much of this research is to provide an alternative way of supporting cooperative work to that found in, for example, workflow systems (e.g. Glance et al., 1996) where work activities are given a formal representation in terms of some workflow model which often stipulates how the contributions of different participants are to be coordinated. In contrast, in many awareness-oriented systems, the coordination between different activities is supported by giving participants an awareness of what each other are doing or have done so that participants can coordinate their work themselves. Many researchers would hope that, not only does this provide a 'truer' and more 'lightweight' sense for 'support', but would also make for more flexible applica-

J Hughes et al. (eds.), Proceedings of the Fifth European Conference on Computer Supported Cooperative Work, 221-236.
© 1997 *Kluwer Academic Publishers. Printed in the Netherlands*

tions which are not liable to the usability criticisms (cf. Bowers et al., 1995) that can be made of more procedural-oriented approaches to CSCW.

Perhaps these arguments find their most detailed elaboration in work on Collaborative Virtual Environments (CVEs), where Virtual Reality (VR) technology is used to support cooperative applications such as virtual conferencing or collaborative information visualization and retrieval. Most notably, the COMIC project offered a 'Spatial Model' of interaction in shared virtual environments (Benford et al., 1994, 1995) which has provided the basis of a number of experimental applications as well as influencing the fundamental architecture of at least two VR systems: DIVE (Carlsson et al., 1992) and MASSIVE (Greenhalgh et al., 1996). The question arises, however, of how this research theme is to be further advanced as a major constituent of CSCW endeavour.

We would wish to argue that some of the most promising work currently on the theme of awareness in CSCW is concerned with one or both of two issues. First, there exist attempts to integrate support for awareness at fundamental levels of cooperative system architecture. We have already mentioned how the DIVE and MASSIVE VR systems implement awareness-oriented notions. Trevor et al. (1994) report on how a shared object service can be designed to facilitate user's awareness to the state of and changes in shared objects. Further attempts to 'build in' awareness as a foundational feature of cooperative systems are likely to be seen. Secondly, some of the concepts, models and notations elaborated in the development of awareness-oriented applications and systems will be found to have a broader utility. As an example of this, see Rodden's (1996) work showing how support for awareness can be added to workflow, shared databases and other cooperative systems.

In this paper, we attempt to address both these issues. First we show how concepts derived from the COMIC Spatial Model (cf. Benford et al., 1994, 1995) can be reinterpreted to have general utility beyond the domain of shared virtual environments which was their initial application. We present how we use the model and the new concepts we introduce. We describe the current implementation of an awareness engine, called AETHER, based on the suggested model. Our goal is to recognise awareness at a fundamental system level and to build other functions on top of it. In order to demonstrate the implementation, we present some small applications that we have developed as well as some ideas about how other systems can be implemented. The paper ends with plans for future development as well as drawing out the general implications of our work for CSCW research.

The Spatial Model

As the Spatial Model, largely developed in the European Communities' COMIC project (1992-1995), forms the basis for our work, we will spend a little time describing its essential elements. The Spatial Model supposes that objects (which might represent people, information or other computer artefacts) can be regarded as situated and manipulable in some space. The notion of space is very generally conceived only subject to the constraint that well-defined metrics for measuring posi-

tion and orientation across a set of dimensions can be found. In principle, any application where objects can be regarded as distributed along dimensions such that their position and orientation can be measurably determined is amenable to analysis in terms of the Spatial Model.

The interaction between objects in space is mediated through the relationships obtaining between three subspaces: aura, focus and nimbus. It is assumed that an object will carry with it an aura which, when it sufficiently intersects with the aura of another object, will make it possible for interaction between the objects to take place. On this view, an aura intersection is the pre-condition of further interaction For objects whose aurae intersect, further computations are carried out to determine the awareness levels the objects have of each other. The subspaces of focus and nimbus are intended as representing the spatial extent of an object's 'attention' and its 'presence'. Thus, "if you are an object in space, a simple formulation might be: the more an object is within your focus, the more aware you are of it; the more an object is within your nimbus, the more aware it is of you." and accordingly, "given that interaction has first been enabled through aura collision: The level of awareness that object A has of object B in medium M is some function of A's focus in M and B's nimbus in M." (Benford et al., 1994)

It is important to note that in the above definition, awareness-levels are defined per medium. Thus, the 'shape' and 'size' of each of the aura, focus and nimbus subspaces can be different, for example, in the visual (graphical) than in the audio-medium. In this way, I may be aware of the sounds made by another object but without being able to see it. Benford et al. (1994, 1996) go on to show how simple instantiations of this model can have a high degree of expressive power, for example enabling one to distinguish between different intuitively familiar 'modes of mutual awareness' on the basis of the possible relationships between A's awareness of B and B's awareness of A. However, perhaps the most important point emphasised in this work is the insistence that awareness is a joint-product of how I direct my attention to you (focus) and how you project your presence or activity to me (nimbus). Applications which recognise only one of these two components may well be experienced as too intrusive or too inflexible.

In recent work, various extensions of the Spatial Model have been reported. For example, Benford et. al. (1997) have introduced a concept of 'third party objects' which 'intervene' between objects and transform the nature and level of the awareness that the objects might otherwise have, and, importantly, Rodden (1996) has reinterpreted the Spatial Model in terms of spaces which can be represented as graphs of interconnected objects.

The AETHER Model

Our further development of the Spatial Model and the idea of an 'awareness engine' resulted from our previous studies like @WORK (Tollmar et al., 1996) and related

projects such as CODESK (Tollmar et al., 1995) where different awareness clues for supporting information sharing and casual interaction are provided. CODESK was developed as an open environment where new applications could be added for specific tasks like editing or communication. Other systems have been taking similar approaches, e.g. GROUPDESK (Fuchs et al., 1995) or TEAMROOMS (Roseman et al., 1996). We have been using CODESK as a 'target system' for our awareness engine, so as an introduction to the AETHER model we will present how the awareness engine would relate to the overall architecture of a system like CODESK.

The Structure of the System

We place the awareness engine at a basic level of system architecture (Figure 1). The engine is intended to provide applications with the necessary information about what users are doing or have done.

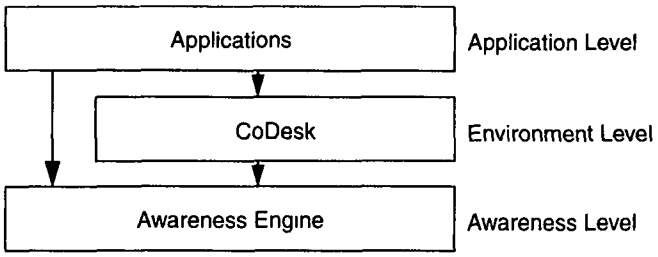

Figure 1 The Place of the Awareness Engine in the Structure of the CODESK system

The second level is that of the environment. This will include all the basic functionality: shared file access, access control, versioning, communication channels, etc. All these functions are to be built on top of the awareness engine.

The top level is the application level where specific awareness information is collated and presented to the user. Applications are written based on the functionality provided by the second level. They can interact with the awareness engine in two ways: by interacting with the environment or by directly accessing the engine.

The Semantic Network

Traditional CSCW systems usually keep a *structural network* of objects. For example, a classical shared file system can be represented as a tree structure by means of the 'containment' relation. Other kinds of relation can have their own representation. For example, ownership may be represented in terms of parameters associated with relevant files or folders. Inspired by systems like GROUPDESK we also integrate representations of users and groups, as well as the result of their actions, into the structure obtaining a *semantic network* that forms a "representation of the working context" (Fuchs et al., 1995). This network made of *objects* interconnected by *directed relations* comprises the space in which awareness computations are done.

The *objects* in the network can be any entity (files, folders, applications, people, groups, sessions, whatever) defined by the environment and its applications. The awareness engine will treat all objects in the same way making no assumptions about the kinds of thing the objects represent. The *relations* that connect the different objects can also be of any type: structural relations (e.g. containment), user interaction relations (e.g. open file), property relations, and so forth. Once again it is up to the environment and its applications to define the specific type of relations. This semantic network creates a *space* in the sense similar to that suggested by Rodden (1996). This network will be the space in which aura, nimbus and focus are defined and in which the awareness levels between the different objects are computed.

Moving Away From Events

In many existing CSCW systems awareness information is obtained by means of *events*. User actions like file access, modifications, etc., are monitored, selected according to some criteria, and eventually recorded as event lists. For example, Fuchs et al.'s GROUPDESK keeps lists of events which are used to provide 'asynchronous awareness' even if some of the information in this event list duplicates information which could be derived from the semantic network. Events are discharged based on some distribution strategies defined in advance. As Fuchs et al. show, such event based systems seem to work satisfactorily for situations where workflow can be clearly defined in advance or if the application is known from the beginning.

However, as we remarked in the Introduction, much of the promise of awareness-oriented CSCW systems lies in their potential to offer a flexible alternative to strictly defined workflow approaches. So it would be somewhat ironic if an awareness mechanism only worked adequately in tandem with a system supporting somewhat rigid workflows. In contrast, systems like @WORK are intended to address groups of users with highly flexible working arrangements. This would make it hard to define appropriate selection criteria and distribution strategies in advance, an argument which is especially telling for systems (like CODESK) which provide an environment for new applications (and hence new user-actions and event-types) to be readily built. Thus, for the systems we are interested in, an event distribution approach for supporting awareness does not seem appropriate.

As an alternative we propose keeping all objects and relations in the net, even after their expiration, and using Spatial Model concepts to determine awareness levels for them. Instead of removing expired objects and relations we mark them as invalid. With this we have no need for event lists because the information that those contain is now in the objects and relations of the network. What we obtain is a semantic network containing both the actual state of the system as well as all history information. Of course, the disadvantage of this approach is the quick growth of the size of the net but later we discuss some ways for reducing this size.

Reformulating 'Time' and 'Medium' Spatially

By keeping the invalid relations in the network, we can compute meaningful awareness information not only about what happens right now but also about past events. In this way we can say that our space equally contains two aspects: the semantic and the temporal. Time now becomes one of the 'dimensions' of the space, the concepts of the Spatial Model equally applying to it as to any other dimension.

A cognate approach can be taken to the notion of a 'medium' of communication. In the original Spatial Model, medium was loosely defined based on an intuitive understanding of this concept or, in Rodden's (1996) generalization, as a label on aura, nimbus and focus. In our case, as the space we have is not geometrical, we found it necessary to devote more attention to this concept. By medium we understand (a) a well defined type of information, (b) a subspace that has the capability to carry that specific information and (c) some objects that 'understand' that medium. Two objects that understand the same medium can communicate though it. Information will be generated by one of the objects, will travel through the medium subspace and will be received by the second object. An analogy is radio, where an antenna transmits an electromagnetic wave that will propagate through all objects, even if these are not sensitive to it, while a radio, which can 'understand' the wave, will convert it to sound. For us, the medium's subspace is made of objects and relations, even if those do not understand that respective medium. Aura, nimbus and focus will be defined per medium subspace.

Medium also has a time component. For example, a medium can define a subspace that contains only those objects and relations that have been valid during some time period. In this way we can obtain a time window. In this approach, a moment can be defined by collapsing the time window. Thus, in a 'synchronous medium' the time moment of 'now' is achieved by filtering out everything that is in the past. The 'synchronous' becomes a subcase of the 'asynchronous'. We suggest that this will facilitate systems to provide smooth transitions between different forms of communication, a point we shall return to at the end of this paper.

Aura, Nimbus and Focus

Aura in our model is much the same as defined in the initial Spatial Model. It describes the potential for collaboration between two objects. If there is sufficient aura intersection (e.g. collision) then there is potential interaction between them. Given our approach to the notion of a 'medium', aura is defined by the medium rather than objects themselves. Nimbus and focus in our system have much the same meaning as in the Spatial Model. Each object or relation can control its focus and nimbus to specify their 'willingness' to become aware of others or fall within their awareness. Given the temporal aspects of medium just argued for, it should be observed that aura, nimbus and focus also have a time component. Thus, a user can 'focus' on the present moment, on the past or even on the future. This is exploited in a prototype versioning system presented later on.

We considered that our engine would be most flexible if both objects and relations in our network could have aura, nimbus and focus. By allowing a relation to have nimbus we allow users to get notified about the presence of a relation. In this way the user is aware not only of objects but also of the activity of others to the extent that this is depicted in the relations and changes to them. As we shall shortly see, however, this necessitates a reconceptualization of how awareness can propagate through our space.

Presence

People and other agents can manifest a presence in the network space. Presence is defined as (a) the agent, (b) an application that the agent uses for manifesting its presence in one or more media and (c) the object where the presence is located in the net. For our purposes, an agent can be a person, a group of people, or a computer agent. We say that an application is present in a medium if it 'understands' that respective medium. The object defining the location is much like Rodden's (1996) definition of presence in non-geometrical spaces. Like him, we allow an agent to be present in more than one place in more than one medium at any given moment.

Medium Consumption

In the initial Spatial Model the level of awareness that an object A has of an object B in medium M is computed, if aura collision exists, as "some function of A's focus on B in M and B's nimbus on A in M" (Benford et al., 1994). That is, the awareness-level is obtained through a negotiation between A and B by means of controlling their respective foci and nimbi. We would like to add to this a new concept: space as an aura, nimbus and focus 'consumer'. Our point is twofold: first, that the level of awareness should not depend only on the two objects but also on the nature of the space between them, and secondly that space cannot be seen as an empty, passive 'container' for aura, nimbus and focus but rather become an actor in 'negotiation' of the awareness levels.

Fog provides a relevant analogy. Fog consumes part of the light and the sound passing through it, filtering out the fine details of objects perceived through a fog-filled subspace. Indeed, fog not only fills a subspace but also comprises a collectivity of very small objects, each one with a specific behaviour and a filtering effect. It is this conception of space as always 'filled' which motivates our choice of name: AETHER.

Accordingly, we redefine the level of awareness that object A has of object B in medium M, in case of aura collision, as being some function of A's focus in M *'filtered by' the space between A and B* and B's nimbus in M *'filtered by' the space between B and A*. This definition necessitates two remarks concerning our notion of spaces consuming focus and the rest. First, consumption is not necessarily symmetrical, that is, the consumption depends on the direction of the information flow. For example, the consumption of nimbus from A to B can be different to the one from

B to A. Second, consumption need not have only a diminishing effect as some elements in a space may also amplify aura, nimbus or focus.

The idea of consumption relates with other concepts of the Spatial Model used for manipulating aura, nimbus and focus. For example adapters (Benford et al., 1994) and third party objects (Benford et al., 1997) are mechanisms used for the same purpose. The main difference is that both concepts use objects for this manipulation. Our model is more general, space itself having this effect on aura, nimbus and focus, with space comprising not just objects but also their relations in a structured semantic network. Objects and relations in our system thus have a double role. First they are the ones manifesting their presence by generating aura, nimbus and focus. At the same time they form the space of the model so they will become consumers of aura, nimbus and focus. As such, any object or relation in our system can be seen, and used, as an adapter or third party object.

The consumption of aura, nimbus and focus also has a meaning in the time dimension. After all, time does have an effect on the importance of objects and relations. For example the importance of a certain user action might decrease in time; it might be important right now or five minutes later but it might have no importance at all in one month.

According to this new definition, the computation of the aura, nimbus and focus becomes a negotiation between the two objects or relations (A and B), the medium that both of them 'understand' (M) and the space between the two. The medium M defines the aurae of A and B while the aura consumption is defined by the objects and relations on the relevant path(s) between them. If the aurae intersect at some point at a high enough level, then focus and nimbus computations will take place. A will define its initial nimbus value, then the different objects and relations on the path(s) between A and B will consume it. B will define the initial value of its focus and again the objects and relations on the path(s) between the place B is focusing on and A will consume it.

The Percolation Mechanism

Before describing the implementation, let us explain how we see aura, nimbus and focus 'permeating the Aether'. Although there are a number of possibilities, we have explored a concept of percolation. Consider a case when the aura associated with an object A percolates from A (its 'source') through the objects and relations that belong to the relevant medium's subspace.The objects and relations which are neighbours of A will each consume the aura to some degree, as will, in turn, the neighbours of these neighbours. And so forth. The process of percolation stops wherever the aura level decreases to some threshold or below (zero in our case).

In Figure 2 we have an example of aura percolation. The numbers next to the different objects show the level of aura reaching it. We can see how the aura of the object labelled 12 is consumed from one object to another, except between objects labelled 9 and 15 where it is amplified. The relations or objects that do not belong to the medium M subspace are not taken into consideration in the percolation.

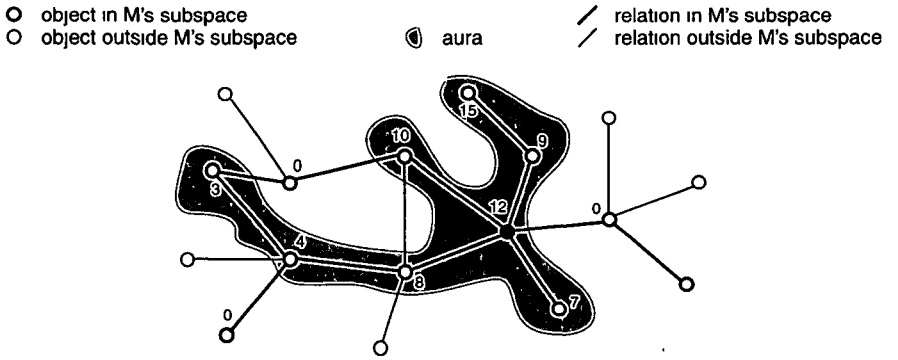

Figure 2 The Percolation of the Aura (or of the Nimbus)

We use the same percolation mechanism for nimbus but, for focus, we 'source' the focus not from the object whose focus it is but from the object(s) or relation(s) on whom focus is directed.

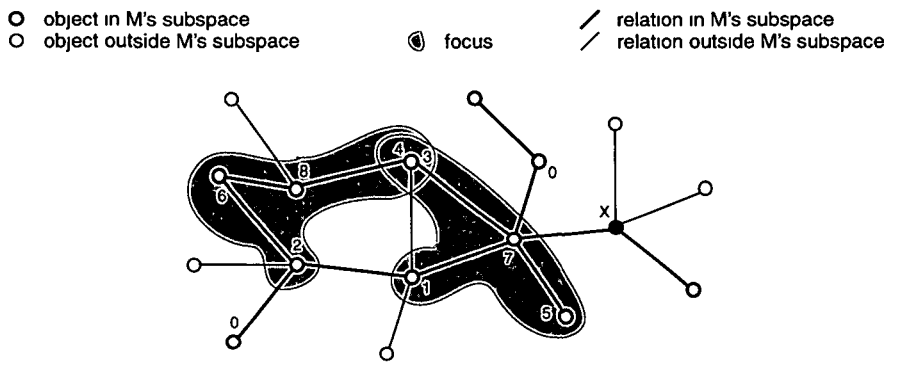

Figure 3 The Percolation of Focus

In Figure 3 the focus of object X is made of two percolations, one centred in object 8 and one in object 7. The use of the percolation for focus is like saying "I am interested in these and what is around them."

Implementation

Currently a running implementation of the AETHER awareness engine is being experimented with. The engine maintains a network of objects and relations, though for reasons of parsimony and convenience, in our implementation the relations are defined as objects as well. Each relation points to two objects, a *from* and a *to* object to define the directionality of the relations. An object can point to none, one or more

relations. In this way, what we have called 'objects' and 'relations' so far can be thought of as specializations of a *component* concept.

A component (object or relation) has a nimbus value and a nimbus 'strategy' attached. The nimbus strategy defines the way nimbus percolation will take place. Each component can also have one or more focus points, each of them with a focus value and a focus strategy. It is up to the environment or any application to set the focus point(s), the nimbus/focus values and to set or modify the nimbus/focus strategies. A percolation strategy, be it for nimbus, focus or aura, is defined as a function that answers the question "is object X part of object Y's nimbus/focus/aura?"

A medium in our implementation has to define (a) the medium's subspace and (b) the aura percolation strategy. The subspace is defined by means of a function that answers the question "is object X in medium M's subspace?". The presence of an agent is implemented by using a presence object. This object is connected to (a) a user, group or software agent by a 'represents' relation, (b) an application by a 'uses' relation and (c) a component in the net that defines the location by a 'visits' relation. Each component also contains the definition of the way in which aura, nimbus and focus are to be consumed. In the case of relations, consumption can vary according to the direction of percolation. Consumptions are defined as strategies that answer the question "how much of the aura/nimbus/focus value will remain?"

As we see, for each object the environment or the applications have to define a number of percolation and consumption strategies. In order to simplify things we have developed a number of basic strategies. These strategies can be logically combined in order to obtain more sophisticated ones. At the same time, any application can define new strategies rather than combine the pre-defined ones. In this way, it is hoped that the engine is both simple to use and flexible. An example of a pre-defined strategy is the 'now' medium subspace. The answer to "is object X in medium M's subspace?" will be "yes, if it is valid".

The Algorithm

Now we can define the awareness level computation algorithm. The computation is done for each medium separately. For a given object or relation (source), the engine starts from its neighbours as the first set of candidates, with a given initial strength. Each candidate is then checked as to whether it included in the medium's subspace, by asking the medium's subspace strategy. If it is, then the aura (defined in the medium), nimbus or focus (defined in the origin object) strategy is asked to confirm that candidate. If it gets accepted then the candidate's consumption strategy is asked to compute the new strength that reached it. If there is some strength left, the candidate becomes part of the computed subspace, having its membership characterized by this strength. Its neighbours will be considered as candidates in the next step. This process continues until no other candidates can be considered.

The awareness level between two objects A and B is defined as four strength values: A's focus on B, A's nimbus to B, B's focus on A, B's nimbus to A. If A's and B's aurae don't have common components (i.e. there is no adequate aura intersec-

tion), these values are null. After all the computations have been done, each presence object will get a vector of all components with whom it has aura collision and the respective awareness levels. As the intention is to provide as much generality as possible, the decision on how to interpret the four values is left to the application in question. One way would be to interpret them according to the 'modes of mutual awareness' as defined in Benford et al. (1994) and Bowers (1993).

The computation is repeated after any change in the network, that is after any user action that affects the state of the system. In order to reduce the data traffic between the engine and the applications, AETHER keeps track of the awareness level between every presence object and the other components. After each re-computation the new awareness levels are compared to the old ones and changes are reported to the applications.

We have defined an Application Programming Interface (API) for the AETHER engine to support communication between it and CODESK or the applications. The engine API is a very simple one, letting applications add, validate or invalidate relations and change strategies, and in the other direction, allowing the engine to announce awareness levels to the applications.

Computational Considerations

It is obvious that the AETHER model will raise issues concerning computing time and network size, especially if relations and objects continue to be stored after their invalidation time. We can address the computational complexity problem in several ways. The engine currently makes use of a number of techniques to reduce the number of computations, for example by performing multiple changes in the net before awareness level recomputation. Parallelization is another possible approach. As calculations in different media (and calculation of nimbus and focus in the same medium) can proceed independently of each other, CPUs can be allocated on a per medium (or per awareness-subspace) basis.

Computation time can also be facilitated by carefully managing the size of the network. In this regard, we suggest that from time to time certain objects and relations can be removed in a process much like garbage collection. The question is which objects and relations are important to maintain and which are of lesser significance and can be removed. Ultimately, 'importance' can only be properly defined at the application level, though we do provide a general technique at the awareness engine level which can be used quite flexibly in default of specific requirements made by the application.

In our model, the importance of a component is defined by its nimbus in time and our garbage collection algorithm periodically applies some consumption of this nimbus in terms of a function which reduces its value according to the distance in time since invalidation. The engine then removes the components whose nimbus falls below a given threshold. The system also removes relations connected to objects that have just been deleted and, if this now leaves objects without relations, then these are deleted too.

While this algorithm seems to work well for the applications we have experimented with, there is clearly scope for refining it. For example, we could relate the importance of an object to the number of times it has been 'visited' by users. An object visited often may be more important than one not visited at all. A visit could have the effect of incrementing the nimbus of the object, thus tending to increase its longevity.

Applying the AETHER Model

In order to demonstrate the feasibility of our approach we have developed a number of small applications. We have primarily concentrated on showing how services often thought to be fundamental to cooperative applications can be readily supported in the AETHER model, in particular, the management of versioning, history and access control. To demonstrate the generality of our model we have also simulated a version of Isaacs et al.'s (1996) PIAZZA prototype awareness system. We briefly discuss these applications in turn.

Versioning and History

Some kind of versioning is normally needed in CSCW applications. We will show one way of implementing it with the awareness engine. Each version of a document (say) is represented as an object in our net. A 'is-previous-version-of' relation binds the different versions into a version tree. A user can access the latest version or can focus on some previous time moment, by selecting the appropriate focus strategy, and access the versions that existed at that moment. Users could also have access to any other information about these documents, like for example who changed them and who's read them, by controlling nimbus and the focus.

In many cases, it is likely that after a while the number of versions would be too big and some would have to be removed. The point would be to remove the minor versions and to keep the important ones. For this, the versioning module would have to set the nimbus of the different versions in such a way that by applying the garbage collection algorithm the desired effect would be obtained. One way to do this would be to relate the level of nimbus in time with how much the new version differs from the previous one. In case of minor changes the nimbus would be small, while extensive changes would generate a high nimbus and would remain in the system for a long time.

History is a related problem but it refers to user actions over time instead of documents. Very many history events can be deduced from the time information of the different relations that represent user actions. By setting the focus strategy to some moment in time the user would be notified about the state of the system at that time around the point of focus. By displaying all the changes in the objects and relations (creation or invalidation) between two moments in time we could reconstruct a history of events. It may be that some components have been removed from the net-

work but, as important events tend to be more long-lasting, this method of reconstructing history should be satisfactory for many purposes.

Access Control

Another important functionality needed in CSCW is access control. An interesting way to do this in the AETHER model is to have access control media. For example we could define a 'Top Secret' medium. The boundaries protecting an area that contains sensitive information could consume completely the aura, nimbus and focus of all other access media, except for the 'Top Secret' one. Only users that are allowed to use this medium would be able to notice the presence of those objects and access them. In this approach, boundaries can be realised by particular kinds of objects in the net which consume aura and the rest and can exert constraints on navigation through the net (cf. Bowers, 1993).

Another interesting approach would be to build on a suggestion in Benford et al. (1996). A 'Foyer' could be used for entering the system. One of the functions of the foyer is to "enhance security by providing a single point of entry... within which incoming and outgoing people are made publicly visible and hence accountable". The system could have such an entry point where all users would have to start and at which their capabilities (or 'strategies' in the sense used above) for manipulating and consuming aura, focus and nimbus would be defined. A guest, to give just one possible example, may have a more limited focus (so that they tend to access less) but a larger nimbus (so that other users are likely to be made aware of them and their activities) than a 'registered user'. As these capabilities can be defined on a per medium basis, a very flexible approach to access control is possible through the AETHER model. We refer to a given profile of awareness manipulation and consumption strategies as a 'character'. While taking on a specified character may be necessary to gain full access to a certain medium subspace, this is much more flexible than traditional approaches which typically give and withhold 'access rights' to 'roles'.

A Simulated PIAZZA

Our final test of the feasibility of the AETHER model and our awareness engine implementation is a 'simulation' of PIAZZA, an application prototyped by Isaacs et al. (1996), which provides users with information concerning "others who are doing similar tasks when they are using their computers, thereby enabling unintended interactions" (p. 315). PIAZZA comprises a number of sub-applications, two of which we have reimplemented using the concepts of the AETHER model: GALLERY which allows the user to get information about other group members, and ENCOUNTER, a component which can be added to any application and which makes users aware of others who may be "nearby" (see Isaacs et al., p. 319-321).

Our GALLERY is an application that sets its focus on the people selected by the user. The application uses a percolation strategy that will define the focus in terms

of those relations around a person that show their current activities. When such activities exist, the application will present to the user what the others are doing and where in the network space they are. Our version of ENCOUNTER is a file browser that, in addition to traditional functionality, informs the user about the presence of others in the same subdirectory of the file system. The application sets the focus around the directory where the user is located and uses a strategy that monitors any other presence in that place. Our treatment of temporal relations as also being part of the network enables us to entertain extensions of Isaacs et al.'s work so that users can become selectively aware of others who have shared the same directory space at different past times. Accordingly, an ENCOUNTER application built upon our awareness engine may be able to support a richer set of "unintended interactions" and social encounters than Isaacs et al. currently discuss.

Discussion

We hope that we have demonstrated the feasibility of the AETHER model as a source of concepts for an awareness engine for CSCW. We believe that several forms of basic functionality for cooperative applications can be readily and flexibly implemented in an AETHER-style engine and have outlined our approach to versioning, history and access control. We have also shown how an existing prototype (Isaacs et al.'s PIAZZA) can be reimplemented (or 'simulated') in the AETHER awareness engine. As our approach can provide flexible solutions for basic CSCW functionality as well as have the capability of simulating other systems, we have a degree of confidence in both the relevance and generality of the AETHER model.

Future Work

In future work, we will investigate further applications of the model. For example, it is easy to see how the basic functionality we have discussed could be combined in, say, a flexible approach to workflow support. Rodden (1996) observes that most workflow systems depend at some level on a graph specifying transitions between states in the workflow. Such graphs can constitute a graph-space over which aura, focus and nimbus can be defined and manipulated. In this way, participants to a workflow can be made aware of activities 'upstream' which are about to become their responsibility as well of activities 'downstream' which follow on from what they have completed. In AETHER, we would add to the graph the documents in their various versions, representations of the users themselves, and any other object or relation of significance, and do so while the workflow is being enacted. In this way, the structure of a workflow can dynamically unfold and be enriched over time, with participants being present in and aware of various subgraphs as determined by the awareness computations. This approach has two attractive consequences. First, workflow graphs are no longer seen as stipulations of the states of the workflow. They become instead 'seeds' for a semantic net which will be added to as the work-

flow unfolds. Indeed, in some implementations, the pre-defined states might even get garbage collected if they are infrequently visited, that is, if they become irrelevant to the way the work has turned out. Secondly, as participants have points of presence within the graphs, which they themselves add to and manipulate their awareness within, the AETHER model could encourage workflow systems which support 'workflow from within', the self-organising and emergent structuring of work in response to contingencies, and not just mandate 'workflow from without', the execution of pre-defined procedures no matter what (cf. Bowers et al., 1995).

Other work on the AETHER model and our implementation of it will be devoted to optimising the performance of the engine by exploring different kinds of percolation and garbage collection algorithm. We will also need to develop methods for 'calibrating' these algorithms for different applications, so that, for example, the engine does indeed identify 'important' components and accord them longer life in a manner which matches user-requirements. Once this has been achieved and an appropriate application developed (perhaps a workflow application within the CODESK environment), a program of user-evaluation will be necessary. We also intend to investigate parallel implementations of our algorithms. Experience with this would also inform further iterations of development with conventional machines (e.g. the appropriate use of background processing for awareness computations). Finally, we intend to explore ways in which the model can be mixed with geometrical models such as those associated with VR systems to obtain hybrids or in order to use geometrical (2D or 3D) interfaces to our model.

Awareness Beyond the Synchronous/Asynchronous Distinction

We want to finish by drawing out a general conclusion for CSCW research from the AETHER model. We remarked above that we treat time as another dimension in constructing the graph 'spaces' over which awareness computations are done, enabling various 'awareness windows' on past events to exist. Equally, by manipulating the form that focus takes, a user can broaden or restrict the extent of objects and relations of potential relevance to their work. This approach enables us to capture within a unified framework all of the forms of awareness in cooperative systems identified by Fuchs et al. (1995): coupled-synchronous (what is *currently* happening in the *actual scope of work*); uncoupled-synchronous (what happens *currently* anywhere else *of importance*); coupled-asynchronous (what happened in the *actual scope of work since the last access*); uncoupled-asynchronous (what happened anywhere else *of importance since the last access*).

'Actual scope' means, in our model, 'being in the focus of the user', 'of importance' means 'the user is in the nimbus of an object or a relation', 'currently' means 'the time focus is now', and 'since the last access' means 'in the time focus between the user's last access and now'. By manipulating the aura, focus and nimbus of the user and of the objects of the system, the awareness engine can generate awareness information for all these situations. But more than this. By translating the coupled-

uncoupled and synchronous-asynchronous distinctions into concepts which admit of continuous variation, we can identify all the 'points in between'.

By offering a framework in which synchronous and asynchronous awareness can be supported equally, we 'deconstruct' this distinction in a unified approach. This is a powerful conclusion because the distinction between synchronous and asynchronous is used so very commonly - often as a way of distinguishing between different kinds of system. While the distinction may be clear at system levels where different communication protocols are discussed, perhaps we should not crudely transpose the distinction so that it classifies different types of awareness, still less different types of cooperative work. What matters to cooperative work *as it is experienced*, we suggest, is the *integration* of different streams of work which may be on many different time scales and show varying degrees of relevance to the matter at hand. Having a level of system architecture where different forms of awareness can all be supported together seems most appropriate to this image. The AETHER model and our experimental awareness engine comprise our attempt at this.

References

Benford, S., Bowers, J., Fahlén, L., Mariani, J. and Rodden, T. (1994): "Supporting Co-operative Work in Virtual Environments", *The Computer Journal*, 37, 8, pp653-668.

Benford, S., Brown, C , Reynard, G and Greenhalgh, C (1996): "Shared Spaces: Transportation, Artificiality, and Spatiality", in *Proc. of CSCW'96*, Boston, ACM Press, pp. 77-86

Bneford, S , Bowers, J., Fahlén, L., Greenhalgh, C., Mariani, J and Rodden, T (1995): "Networked Virtual Reality and Cooperative Work", *Presence*, vol. 4, no. 4, pp. 364-386

Bowers, J. (1993): "Modelling Awareness and Interaction in Virtual Spaces", in *Proc. of the 6th MultiG Workshop*, Stockholm

Bowers, J , Button, G. and Sharrock, W. (1995). "Workflow From Within and Without Technology and Cooperative Work on the Print Industry Shopfloor", in *Proc of ECSCW'95*, Stockholm, Kluwer Academic Publishers, pp. 51-66.

Dourish, P. and Bellotti, V. (1992). "Awareness and Coordination in Shared Workspaces", in *Proc of CSCW'92*, Toronto, ACM Press, pp. 107-114.

Fuchs, L., Pankoke-Babatz, U. and Prinz, W. (1995): "Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System", in *Proc of ECSCW'95*, Stockholm, Kluwer Academic Publishers.

Glance, N., Pagani, D. and Pareschi, R (1996): "Generalized Process Structure Grammars (GPSG) for Flexible Representations of Work", in *Proc. of CSCW'96*, Boston, ACM Press

Greenhalgh, C and Benford, S (1996). "MASSIVE: A Virtual Reality System for Tele-conferencing", *ACM Transactions on Computer Human Interaction*, ACM Press.

Isaacs, E , Tang, J. and Morris, T (1996): "Piazza: A Desktop Environment Supporting Impromptu and Planned Interactions", in *Proc. of CSCW'96*, Boston, ACM Press, pp. 315-324.

Rodden, T (1996). "Populating the Application A Model of Awareness for Cooperative Applications", in *Proc of CSCW'96*, Boston, ACM Press, pp. 87-96.

Roseman, M. and Greenberg, S (1996): "TeamRooms: Network Places for Collaboration", in *Proc of CSCW'96*, Boston, ACM Press, pp. 325-333

Tollmar, K. and Sundblad, Y. (1995): "The Design and Building of the Graphical User Interface for the Collaborative Desktop", *Computer and Graphics*, vol 19, no. 2, 1995.

Tollmar, K , Sandor, O. and Shömer, A.(1996): "Supporting Social Awareness @Work - Design and Experience", in *Proc. of CSCW'96*, Boston, ACM Press, pp. 298-307.

Trevor, J , Rodden, T. and Mariani, J. (1994): "The Use of Adapters to Support Cooperative Sharing", in *Proc of CSCW'94*, Chapel Hill, ACM Press, pp. 219-230.