7. SUMMARY

This paper has presented a general model for machine-generated representations of information useful for awareness. This model uses logging information about user activities, and has a number of interesting properties relative to other awareness schemes.

One of the strengths of the activity-based approach is that it makes awareness information available in a format that is easily machine-parsable and -storable, unlike some other media, such as video. This trait makes activity-based awareness useful as input to applications, in addition to as input to users, which has been the traditional use of machine-augmented awareness.

We have presented an implementation model that can be used to capture, store, and disseminate awareness information. This model captures activity as a set of tuples representing users, their actions, and the objects of those actions. The object framework, in particular, is extremely flexible and can represent a wide array of data types useful to either application-dependent or -independent awareness.

Finally, we have compared the activity-based approach to a number of other schemes, including the awareness information we are presented with in the physical world. The information available in the activity-based model can be leveraged by other approaches to awareness to "fill in the gaps" in the information presented to users and applications.

REFERENCES

Bly, S.A., Harrison, S.R., and Irwin, S. (1993). "Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment." *Communications of the ACM*, 36: 1.

Curtis, P., Dixon, M. Frederick, R., and Nichols, D. (1995). "The Jupiter Audio/Video Architecture: Secure Multimedia in Network Places." *Proc., Third International Multimedia Conference*, ACM.

Edwards, W.K. (1994). "Session Management in Collaborative Applications." *Proc., Conference on Computer-Supported Cooperative Work (CWCW'94)* , ACM.

Edwards, W.K. (1995). *Coordination Infrastructure in Collaborative Systems*. Ph.D. Dissertation, Georgia Institute of Technology.

Edwards W.K. (1996). "Policies and Roles in Collaborative Applications." *Proc. Conference on Computer-Supported Cooperative Work (CSCW'96)*, ACM..

Fish R.S., Kraut, R.E., Chalfonte, B.L. (1990). "The VideoWindow System in Informal Communications." *Proc. Conference on Computer-Supported Cooperative Work (CSCW'90)*, ACM.

Grudin, J. (1988). "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces." *Proc., Conference on Computer-Supported Cooperative Work (CSCW'88)*, ACM.

Gutwin, C., and Greenberg, S. (1996). "Workspace Awareness for Groupware," *Proc., Conference on Computer-Human Interaction (CHI'96)*, ACM.

Manandhar, S. (1991). "Activity Server: You Can Run But You Can't Hide." *Proc., USENIX Conference*, USENIX Association.

Mantei, M.M, Baecker, R.M., Sellen, A.J., Buxton, W.A.S., Milligan, T., Wellman, B. (1991) "Experiences in the Use of a Media Space." *Proc., Conf. on Computer-Human Interaction (CHI'91)*, ACM.

McGuffin L, Olson, G. (1992). "ShrEdit: A Shared Electronic Workspace," CSMIL Technical Report, Cog. Sci. and Machine Intelligence Lab., U. of Michigan.

Newman, W., Eldridge, M., Lamming, M. (1991). "Pepsys: Generating Autobiographies by Automatic Tracking," *Proc. European Conference on Computer-Supported Cooperative Work (ECSCW).*

Neuwirth, C. M., Kaufer, D. S., Chandhok, R., and Morris, J. (1990). "Issues in the Design of Computer Support for Co-authoring and Commenting." *Proc., Conference on Computer-Supported Cooperative Work (CSCW'90)*, ACM.

Root, R.W. (1988). "Design of a Multi-Media Vehicle for Social Browsing," *Proc., Conference on Computer-Supported Cooperative Work (CSCW'88)*, ACM.

Rouncefield, M, Hughes, J.A., Rodden, T., and Viller, S. (1994). "Working with Constant Interruption: CSCW and the Small Office." *Proc. Conference on Computer Supported Cooperative Work (CSCW'94)*, ACM.

Smith, I., and Hudson, S. (1995). "Applying Cryptographic Techniques to Problems in Media Space Security." In *Proc., Conference on Organizational Computing Systems (COOCS'95)*, ACM.

Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). "The Active Badge Location System." *ACM Transactions on Information Systems*, 10:1, Jan. 1992.

"coverage" and more robust solutions may be obtained. The activity approach supports integration through the use of "controller" applications that directly modify the resource database whenever some condition occurs. In essence, controllers act as proxies on behalf of non-Intermezzo-aware applications. For example, an "active badge controller" can update the location slots of subject resources with physical position information.

6. NOTES ON AWARENESS

As we have seen, the activity-based approach to awareness is capable of bringing a wealth of information into the application domain so that it can be presented to users. There are other interesting repercussions implicit in the activity-based approach that may not be immediately obvious however. This section investigates these.

6.1. Input to Applications, Not Just Users

In most awareness systems to-date, awareness data has *typically* been used directly by users. For example, in most mediaspaces, activity is not captured in a machine-readable form. Rather, it is displayed as video and left for humans to interpret. The salient components of awareness are not directly represented in any computer-based form; they must be extracted by users from the video.

A number of "activity monitor" applications have been built that are similar in principle to the awareness approach used by Intermezzo (Manandhar, 1991, and Newman, 1991), but still these systems generate data for direct human consumption only. Typically the activity data is sent to a client application which presents it in a window on the user's desktop.

In contrast, one of the goals of the activity-based approach used by Intermezzo is to make it possible for applications to behave intelligently in the presence of information about user activity. That is, just as users are aware of the presence and activity of other humans, so can applications be made aware of their users.

As an example, consider a computer-based teleconferencing application that could be made aware when one of its users is on the phone. The application could mute the audio from the conference, and notify other participants that the user is engaged in another task that demands auditory attention.

As another example, consider a collaborative editing system in which a user invites another to join the session. The editing tool could be written to analyze the potential collaborator's location and tasks in an effort to determine if the recipient should be disturbed, or if the request for collaboration should be delayed.

Further, the collaborative toolkit itself can use awareness information to enhance its functionality. Intermezzo uses awareness data as an input to regulate its policy (Edwards, 1996), access control, and session management mechanisms (Edwards, 1994).

6.2. Policy and Access Control is Essential

One obvious concern about bringing awareness into the environment is that the availability of such information is potentially a huge privacy violation. Capturing and potentially storing detailed information about the whereabouts and activities of users requires provisions for strong security guarantees about access to this information.

Acknowledgment of the security risks inherent in awareness systems has been noted many times, particularly in the case of mediaspace systems (Smith, 1995). In almost all mediaspace studies, there are a number of potential users who refuse to use the system even though (a) there is strong physical security (users can simply cover the camera), and (b) storage and machine processing of mediaspace data is not feasible with current technology.

Certainly the security concerns present in mediaspace systems are present in an activity-based awareness system. If anything, the concerns are greater since a machine-parsable representation of the information is readily available. Thus, security and access control schemes are of paramount importance in a coordination support system that makes awareness data available to users and applications. Intermezzo uses a novel approach in which the awareness system itself is used to regulate access control to awareness data objects. This scheme is discussed in detail in (Edwards, 1995).

6.3. Persistence Adds Value

A third point to make about activity-based modeling is that persistence can significantly increase the power of the system. The activity-based approach is particularly useful for capturing the real-time aspects of awareness: the instantaneous condition of the collaborative world and the users in it. The ability to support some persistent data (subjects, for example) is quite useful, and persistence has other uses that have not been fully explored by this research. Two obvious uses are the collection of user statistics, and a form of organizational memory "on the cheap" (by remembering who within an organization has written to a document, for example).

be able to capture some of these complex user states, although even they still have limitations, due to artifacts in the quality of video and audio for example.

A final limitation is that the computer must still decide how to present and use awareness information once it has been collected. In the real world, collection and presentation of awareness information are synonymous, at least when performed first-hand: a user sees a coworker and is instantly aware of a wealth of information. The Intermezzo model only specifies a structure for the representation of awareness data. It does not specify how this information will be presented, or even when or if it will be presented. Again, applications must make intelligent decisions about the presentation of awareness for the information to be usable at all.

5.2. Comparison with Other Computer-Based Approaches

Comparison with other computer-mediated awareness systems is also useful. While the activity-based approach has certain limitations and benefits with respect to "real world" awareness, we shall see that it has a different set of strengths and weaknesses relative to other computer-based schemes.

Two common approaches to computer-supported awareness have been investigated. The first and most common is awareness through computer-mediated audio and video. The second is the use of active badges. Each approach has its own strengths and weaknesses relative to the activity-based scheme, and yet these approaches can be integrated together to complement one another.

Video monitoring systems, generally characterized as mediaspaces, support awareness of the locations and actions of others through a near-constant video presence (Cavecat (Mantei,1991), and so on). Some go beyond the simple connected-office approach to provide a locality-based metaphor for interaction (Cruiser (Root, 1988), Jupiter (Curtis, 1993)). In these systems, awareness is regulated and filtered by co-presence in a virtual space.

Video-based systems, rather obviously, provide high-quality indications of events and situations in the physical world: physical location, physical tasks, facial expressions, and so forth. Such systems, however, provide few if any cues about computer-based tasks and activities. Information about the virtual world can only be inferred from physical side effects.

Another difference is that the awareness information received from a video monitoring system is readily available to humans, but difficult to mechanically parse and store. Activity-based approaches have an opposite set of trade-offs. They provide information that is easily machine-representable, but applications must bear the burden of parsing and presenting the data to users.

The second common computer-based approach to awareness is the use of active badges (Want, 1992). Active badges are devices worn by users which update a global location database as the position of the wearer changes. Rooms in the work area must be equipped with sensor devices to receive the data transmitted by the badges.

Information about physical location and co-presence is the only *direct* information provided by active badges. Other types of awareness information must be inferred from location. For example, the location of a coworker in a boss's office may be an indication of interruptibility.

An advantage of active badge systems over video monitoring is that the information is available in a convenient machine-parsable format.

5.3. Summary of Comparisons

A summary of the three computer-based awareness approaches, video monitoring, active badges, and activity-based, as well as "real world" awareness, is presented in Table 3 below. The last row in the table indicates how amenable the approach is to automated machine-parsing. Some approaches generate data that is more easily represented and stored by computer; the aspect of parsability will be discussed more fully in the next section.

	Activity Based	**Active Badges**	**Video Monitor**	**"Real World"**
Location	◐	●	●	●
Tasks	●	○	◐	●
Interrupts	◐	○	◐	●
Co-pres	◐	●	●	●
Parsable?	●	●	○	○

● = Supported

○ = Not Supported

◐ = Partially Supported

Table 3: Comparison of Awareness Approaches

Together, the strengths and weaknesses of the activity-based approach along with those of other approaches suggest that an augmentation may be effective—by combining multiple approaches better

directory of the file, as well as the file itself, will be generally useful to a number of applications.

Other layers will be domain-dependent. Practically anything below the granularity of "file" will only be understood by applications that have the domain knowledge to interpret the object. For example, the notion of figures in a drawing only has meaning to drawing applications; other tools do not have the domain knowledge or infrastructure to interpret such constructs.

The Intermezzo model represents objects as a list of resources that model the particular artifact at a range of granularities. The toolkit supports several domain-independent layers, namely files, directories, and hosts. Applications are responsible for supporting new layers of the artifact hierarchy that represent domain-dependent views of the artifact—the toolkit *cannot* provide this support, since it is domain-dependent.

Table 2 shows an example of an object list representing the hierarchy used to model a particular artifact, in this case a figure in a drawing. The first column denotes the artifact, viewed at a particular granularity. The second column provides a description of the artifact viewed at this granularity. The third column represents whether the view is domain-dependent or domain-independent. The example shown here uses a number of layers to represent the artifact: the network domain containing the host the artifact resides on, the host itself, the components of the path leading to the file containing the figure, the document containing the figure, and the figure itself.

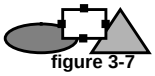
Artifact	Description	Domain
	Network domain of host.	Environment
	Host fileserver	Environment
	Directory path.	Environment
	Document.	Environment
	Figure within file.	Application

Table 2: Example of Object Hierarchy

Note that there are any number of possible additional layers in this hierarchy. Coarser-grained interpretations are possible, as are domain-specific intermediary layers representing constructs such workgroups and projects.

The hierarchical representation of artifacts provides information needed by applications to interpret user activity. Applications can decide at what granularity a given artifact is relevant to them, and interpret the artifact accordingly.

5. EVALUATION

5.1. Comparison with the "Real World"

Comparing the activity-based approach to awareness with the non-machine augmented, "everyday" awareness of others that we commonly experience is instructive, because the comparison offers a common point of reference, and can help identify deficiencies in the activity-based approach.

The most problematic limitation with the activity-based approach is that information is not "free for the taking." In the real world, users simply look around to gather a rich array of information about user activities, locations, and so forth. Humans are adept at determining interruptibility, and other socially-influenced aspects of awareness. As Rouncefield, *et al.*, (Rouncefield, 1994) state, the "ecology of the office provides, to those who know it, the 'at-a-glance' availability of what people are doing, what state they are at, how quickly they are getting the work done, and so on."

In comparison, the activity-based approach requires extensive application cooperation to function effectively. In fact, awareness is one of those situations in which nearly total "buy-in" is required for maximum utility. Similar effects are seen in group calendar tools, where virtually everyone in a work group must use the system without fail for it to function at all (Grudin, 1988). With the Intermezzo approach, applications must cooperate with the runtime system to facilitate awareness.

This requirement for application cooperation is not present in some other computer-based awareness systems, notably mediaspaces, where there is only one stand-alone application that must be run to enable awareness. Users do not have to change their behavior to take part in the system, since mediaspaces essentially "transplant" the remote physical space, along with its associated cues, to the user.

A second important limitation is that some user states are not well-captured by activity. For example, user moods cannot be modeled by this approach. Physical location must be inferred from where applications are being run. Systems that are more like mediaspaces may

Object Resources

Objects represent the focus of a task: the "thing" on which the task is operating. Objects can represent an essentially infinite set of entities on which applications can operate. Common types of objects include files, calendars, database selections, and so forth.

Because of the need for flexibility in representation, objects are more complex than either subjects or verbs For any given task, Intermezzo must be able to generate an object instance that uniquely identifies the thing on which the task is operating, no matter what the type of that thing may be.

We require objects to have the property that there is a one-to-one mapping between an instance of an object resource and the physical entity to which it refers. That is, given an object, one can determine the physical thing it is modeling; and given any physical artifact, it is possible to generate a unique object for it.

The Intermezzo toolkit provides support for objects in the form of a framework for generation, comparison, and retrieval of objects. Individual applications must implement code to generate unique objects for the particular data types on which they operate. Since the object of a given application is domain-dependent, Intermezzo cannot know *generally* how to interpret a given artifact, or even how to generate a one-to-one-mapping to an object instance.

Intermezzo does provide support for generating objects for the most common artifact type: files. The system can uniquely generate objects for any file anywhere in the network. Note, however, that because of application requirements, even the interpretation of files varies from application to application. Developers are free to interpose their own interpretations into the object generation sub system. The section "Granularity and Hierarchy" has more information on defining objects.

Many of the slots in objects are domain-dependent, based on the type of the object. Hence conventions about the definition of these slots are left to the applications that use them. The generic framework established by Intermezzo does enforce a set of slots that are present in all objects and are used for generation and comparison.

The "namespace" of an object defines a particular type of object. For example, files, calendars, and database selections are all represented by different namespaces. Within a given namespace, the name slot uniquely identifies the object as distinct from "siblings" that share its namespace (that is, they are different instances of things with the same type).

The common slots in the generic object framework are represented in Table 1 below.

Attribute	Description
Namespace	The type of the object, represented as a string.
Name	A domain-dependent identifier for the object, valid within its given namespace.
Printable Name	A human-readable string which can be used to identify the object.
Activities	Back-links to the activities containing this object resource.

Table 1: Object Resource Attributes

Comparison of object resources requires comparison of both the namespace and the name. The namespace ensures that no "apples and oranges" comparisons are performed (files and calendars for example); names ensure distinction within a particular type of object ("file1" and "file2" for example).

4. GRANULARITY AND HIERARCHY

One consideration when generating a resource that represents the focus of a task is that applications differ in what aspect of that focus is relevant. For example, consider a user editing a file with a drawing tool. Applications may consider any of a number of characteristics of the object (the file) salient. A project manager tool may need to track when changes are made to any files within a certain directory. Tools specifically built for collaboration may have an even finer-grained focus: a collaborative drawing tool may consider the particular figure within the file currently being manipulated a salient feature of awareness.

Because of the fact that different applications may consider any number of attributes of an artifact salient, simply representing objects as single entities is insufficient. We need a scheme that represents the artifact at a number of granularities: from the machine the artifact resides on, to any containers the artifact may be embedded in (directories in the case of files), to individual components of the artifact that represent the user's current focus (such as figures in a drawing). Objects are represented as hierarchies, with each layer in the hierarchy providing a *view* of the object at a given granularity. As we descend the hierarchy, features hidden in the "large-scale" view become apparent.

In many cases, layers in the hierarchy will be domain-independent. For example, many applications manipulate files. An infrastructure that can generate objects representing the host the file resides on, and the

resources, which represent a space that may be searched using sophisticated search operations (see). Figure 1 shows an application running on a client machine publishing an activity record on the Intermezzo server.

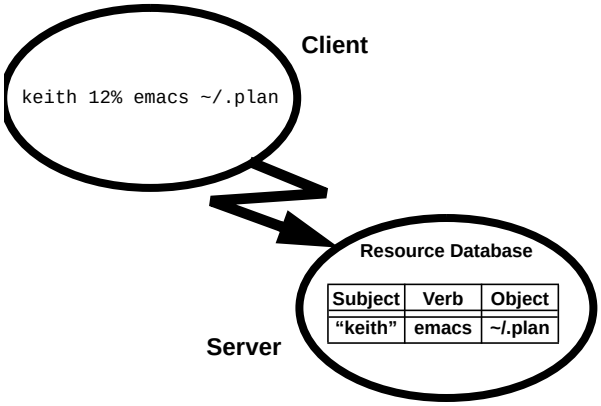


Figure 1: An Application Publishes an Activity Record

3.2. Activity Components in Detail

In large part, the degree to which the Intermezzo model satisfies the facets of awareness mentioned in Section 2 depends on two factors:

- The information associated with each resource.
- The conventions followed by applications and users for publishing activity records.

By pushing a large amount of information about user activity into the environment, we can satisfy a wider range of awareness needs. A richer array of information means that more applications can benefit from the data collected by the awareness system. This section details the information that Intermezzo stores in the global environment as represented by the resources used by the awareness system.

Activity Resources

The Activity resource captures the notion of a single user engaged in a single task on some particular artifact. Activity resources serve as containers for the individual components of an activity: Subjects, Verbs, and Objects. The individual attributes of an Activity resource are links to instances of these resource types, which are described in more detail below.

Subject Resources

Subject resources represent single users of the system. The slots in a subject resource constitute a “single-source” repository of information about a user, and can be used by a variety of applications. Applications are free to access and, given authorization, update slots in subjects.

The slots present in the resource are extensible—applications that need to associate new data with users can add slots as needed. By convention a number of slots are always present—their availability is guaranteed when Intermezzo generates subject resources.

Subject slots include general identifying information such as name, email address, and so on. Subjects also include “back links” to the activities that contain it. From the activity resource, current tasks and objects may be derived. Thus it is possible to generate a complete list of current applications and open files from a given user.

Note that subject resources represents fairly static data. With the exception of the activities slot which, by convention, is updated whenever new activities are created, the data maintained by subjects represent rarely-changed attributes of a given user. For this reason, subject resources are persistent by default (they are long-lived, and survive the termination of any given session or user program). Without persistence it is impossible to refer to a user who is not engaged in any activities.

Other resources in the activity record—verbs and objects—are not persistent by default, since they represent transient states. Human beings, in contrast, are by-and-large long-lived entities, at least in the time span represented by computer applications.

Verb Resources

Verbs represent single tasks. A user engaged in multiple tasks will have multiple activity records, each containing a unique verb representing the task. Multiple verbs will be instantiated if the user is running multiple instances of the same task.

Verbs capture information associated with a single active process in the system. Like subjects, the slots associated with verbs can be extended by applications. A set of common slots are agreed upon by convention and implemented by Intermezzo. Common slots include the application name, start time, idle time, location (hostname), and a back link to the activity resource containing the verb.

Some of this information is static and can be determined at application start-up time (application name, start time, location). Other information, namely idle time, requires application intervention to maintain.

Both the sheer volume and the richness of prior work illustrate the perceived value of augmented awareness by the research community. Perhaps more importantly, the reported experiences of users when participating in such systems is indicative of their value—users *like* working with these awareness systems (Fish, 1990).

This paper presents a model of human awareness based on *activity monitoring* which is useful in either application-dependent or -independent settings. Activity-based awareness can provide a wealth of useful information for coordination. Further, it can deliver information that is complimentary to other approaches. This model has been implemented using the Intermezzo collaboration support environment (Edwards, 1995). We present a systematic format for representing activity, examine the utility of this approach to awareness, and compare it to other awareness schemes.

2. AWARENESS THROUGH ACTIVITY

Gutwin and Greenberg report in (Gutwin, 1996) a set of facets of awareness that should be considered when developing computer-augmented awareness systems:

- *Presence*. Who is participating in the activity?
- *Location*. Where are they working?
- *Activity Level*. How active are they in the workspace?
- *Actions*. What are they doing?
- *Intentions*. What will they do next?
- *Changes*. What changes are they making and where?
- *Objects*. What objects are they using?
- *Extends*. What can they see? How far can they reach?
- *Abilities*. What can they do?
- *Sphere of Influence*. Where can they make changes?
- *Expectations*. What do they need me to do next?

The activity-based awareness approach used by Intermezzo can facilitate a number of these components of awareness. In particular, this approach provides information about presence, location, activity level, actions, changes, and objects.

This work presents a model in which applications cooperate to capture activity information. That information is then represented in a form that makes it readily available to other applications—and to users via applications—that need awareness of others.

3. AWARENESS IN INTERMEZZO

This research takes the approach of promoting awareness by modeling user activities. Activities are represented as a tuple called an *activity record*:

$$A_n = \{S_n, V_n, O_n\}$$

S is the *Subject*, or the user involved in the activity. *V* is the *Verb*, or the current task the user is engaged in. *O* is the *Object*, or the "thing" being operated on. The activity record tuple captures the notion of one user engaged in a single particular task, on an object or set of objects.

This approach has a number of important benefits:

- Information can be collected by a relatively minor set of extensions to applications or toolkits.
- Activity-based awareness satisfies many of the facets of awareness.
- Activity information is easily parsed and stored, and access can be controlled at a fine granularity.

This model has been implemented and explored using the Intermezzo collaboration support environment.

3.1. Implementation

Intermezzo presents to application writers a simple replicated object store. Intermezzo objects, called *resources*, can be shared, searched, and protected via access control lists. A server process coordinates access to shared resources from client applications. The abstract activity model presented in this paper has been implemented directly atop the Intermezzo object foundation: subject, verb, and objects are represented as types of Intermezzo resources. Each of these types has its own set of *slots* (member data) that maintain information relevant to instances of that type.

As we shall see, the generation of subjects and verbs is quite simple. Reliable and semantically-meaningful generation of objects, on the other hand, requires much support from the runtime environment.

By convention, all Intermezzo-aware applications publish activity records when they run. Thus, applications are responsible for keeping the "picture" of the user up-to-date. Users are known in terms of their activities, and the objects of those activities. In essence, users are modeled in terms of their behaviors.

Consider an example: the user "keith" edits a file using an Intermezzo-aware editor. The editor retrieves the information needed to construct resources representing the user, the application, and the edited file. These resources are combined into an activity record and is published to the Intermezzo runtime service. Once published, the activity record and its components are available to any application that has the proper authorization to examine them.

The global "context" in which collaborative applications are operating comprise a number of

Representing Activity in Collaborative Systems

W. Keith Edwards

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
+1-415-812-4405
kedwards@parc.xerox.com

ABSTRACT. Awareness is an important concept in coordination and collaboration at large. Based on prior research in awareness for collaboration, this work presents a data model for representing information about user activity. Fine-grained activity information can be an important tool for facilitating awareness in workgroups. Further, unlike some forms of information used for awareness, activity data is easily stored and parsed. This representational model is implemented atop the Intermezzo collaboration framework and provides a systematic approach to capturing information about users spread across a network.

KEYWORDS: computer-supported cooperative work, awareness, coordination, Intermezzo.

1. INTRODUCTION AND PRIOR ART

Collaborative software, by definition, involves the interaction of multiple people working together to accomplish some task. In most current collaborative settings, these users are likely to be distributed in space or, in the case of asynchronous systems, even distributed in time. To interact more effectively, users need to be aware of others: their presence, actions, and so forth. The goal of awareness is to promote *coordination* among users: making their interactions more efficient and fluid.

In the "real world," users maintain an awareness of others through a number of sources: peripheral sounds, quick glances, information passed along from others. In an attempt to provide some of the benefits of real-world awareness in the computer domain, a number of computer-augmented approaches have proven useful for promoting awareness information among participants and potential participants in a collaborative endeavor.

Computer-mediated awareness can be roughly categorized into two forms: awareness in *application-dependent* and *application-independent* contexts.

In the application-dependent domain, programs must convey a sense of the actions of users within the application: what objects are they manipulating, and what actions are they taking, in the space of objects and actions provided by the application. Examples of systems that provide this form of awareness include ShrEdit (McGuffin, 1992), and PREP (Neuwirth, 1990). For example, in the ShrEdit shared editor, users are presented with a "Gestalt" view, which shows the (virtual) location of participants in the editing session, and highlights their foci in the document.

In the application-independent domain, awareness systems try to convey a sense of the user's place within the world, instead of the user's place within an application. The goals of this form of awareness closely resemble those of "real world" awareness: provide information about location and activity to support coordination, and be useful in a pan-application context. Most current examples of application-independent awareness are *mediaspace*-like (Bly, 1993) systems in which the computer is used as a facilitating device for propagating video from cameras mounted in the physical work space. We can see and hear the context of users in the physical world, and these real-world cues are translated into the computer-mediated world.