ThreeDness: representing awareness in cooperative applications

Fabrizio NUNNARI
Department of Informatics
University of Torino, Corso Svizzera 185, 10148 Torino, Italy

Carla SIMONE
Department of Informatics Systems and Communication
University of Milano-Bicocca, via Bicocca degli Arcimboldi 8, 20126 Milano, Italy

Abstract The paper presents an approach and an architecture supporting the representation of awareness information in cooperative applications. The approach is based on the choice to make a clear distinction between the representation of actors current focus and of the various degrees of awareness information. The architecture is influenced by the basic choice to support flexibility and adaptability of the representation according to users needs. The paper illustrates the approach through a prototype that serves as a demonstrator and discusses the improvements that have been derived from this experience.

1 Motivations and background

The recent evolution of cooperative applications shows a great improvement in their functionality, interoperability and integration with WEB technologies. Unfortunately, the design of their interfaces does not show, in general, the same degree of innovation. Most of the application interfaces are still based on a paradigm based on windows, buttons, etc., which is often adopted in WEB interfaces too. Of course, these considerations can be done for single-user application interfaces but the limits of the above paradigm are especially evident in the case of cooperative applications, at least for two reasons. First, the distributed nature of cooperation increases the complexity of the type of information that has to be represented. In fact, users are not only involved in several activities at the same time (like in single-user applications) but they are also part of several cooperative environments in order to perform them. This means that the user interface has to take into account the co-existence of these environments not only from the point of view of their specialized functionality and data exchange but also from the point of view of the co-existence of several groups of users cooperating within or across these environments. This brings to the second source of complexity, namely the representation of the information that is fundamental for an effective articulation of those cooperative activities. It is now recognized that effectiveness of cooperation is obtained by a mix of coordination supports (like protocols and coordinative artifacts) and awareness of the state of the cooperative work arrangement and field of work. Both coordinative supports and awareness information have to be represented in the interface. In [1, 13] we stressed that this representation has to be integrated as the related supportive technologies should be. In [2]

we proposed conceptual tools towards this integration, with a special emphasis on protocols. However, these tools are useful to elaborate coordination and awareness information only, and leave completely out any aspect of their representation. The latter are instead the focus of the present paper.

The innovation of the cooperative application interfaces is mainly based on the shift from desk-top like metaphors to metaphors describing aspects of the broader space in which co-operation happens. These aspects concern the presence of the involved actors, their location, their availability, their current activities, and so on. Hence, it is natural to look at technologies able to represent the space of cooperation and take into account the above mentioned aspects. For example, WEB-based interfaces can be bent to transform the typical navigation into a journey in an emulated three-dimensional world. Typically, the discrete clicking on active cues (like icons and buttons) is replaced by the continuous displacement of the mouse on the screen to get a more detailed visualization of some parts of the space; or the combination of animations and images represent the change of perspective during the journey. The obvious alternative is to use 3D technologies to simulate the space trough a virtual representation. Since many years there is a lot of researches in the field of VR (Virtual Reality) whose main goal is to reproduce as much as possible realistic environments through computer graphics. With the recent phenomenon of Internet technology, many approaches have been attempted to combine the VR technologies with CSCW principles bringing to what has been defined CVE (Collaborative virtual environment). The main idea is to create a shared virtual world in which several users, represented through avatars, can meet and interact with each other via multi-modal communication [16].

This type of technology has been recently applied, with suggestive results, to interpret artistic performances as the result of a collaboration among real and virtual characters, members of the audience and a production crew [3]. In general, CVE applications are based on scenarios characterized by a tight collaboration among the involved actors in the virtual world. A crucial issue in this respect is the relation between this collaboration and the concomitant collaboration in the real world. In other words, the general question is: how can people be simultaneously immersed in the two worlds? How can the latter be coherent? Another field of application of special types of CVEs is collaborative engineering: here, for example, the possibility to reproduce detailed features of products allows designers to perform distributed simulations in various phases of the design process, with the obvious advantage to collaboratively anticipate and solve emerging problems.

It not immediately evident how the outcomes of the above experiences can highlight the design of interfaces for less specialized CSCW applications. We believe that each situation requires ad hoc considerations, and that it is worthwhile to proceed in different directions before trying any generalization.

The scenario we are taking into account is characterized by the presence of distributed autonomous actors, cooperating in various kinds of activities. Here, distributedness and autonomy imply the articulation of local, autonomous choices to focus on different, concurrently executable activities. In this scenario, cooperative actors do not primarily need full immersion in, nor exacerbated realism of, the representation of the common working space. As for the first aspect the interface should give cooperative actors the possibility to distinguish their currently focused working space from the space generating awareness of the rest of the collaborative environment. As for the second aspect, the representation of the latter has to reproduce the relevant information in a way that it is easy to grasp and understand. Hence, altogether, our approach is closer to the one proposed in Nessie [6] and Tower [7] where the focused working space is kept separated from the awareness space. However, our

approach has some differences. First, we do not aim, at this stage, to 'facilitate encounters' although this feature could be part of our goals in the future. Second, the interface has to give cooperative actors a view of the awareness space from their individual perspective. Finally, the collaborative environment is represented on the basis of spatial metaphors that flexibly combine features of the physical setting with features characterizing the logical structure of the collaborative environment.

Accordingly, we want to use the capabilities of the technologies constructing virtual environments to create Virtually Augmented Common Working Spaces, revisiting the ideas underlying the approach called Augmented Reality, whose aim is to populate and enrich a real world with computer generated graphics. An additional, long-term goal is to realize those spaces so that users can specify them according to their preferences in full integration with the technology they use to support cooperation.

The paper is organized as follows: the next section illustrates the above goals more in detail. Then the approach is exemplified in Section 3 through the description of an implemented demonstrator. Section 4 illustrates how the new architecture goes in the direction of the abovementioned goals. The concluding section identifies the next steps of this research effort.

2 Basic principles and key concepts

The goals mentioned in the previous section require a more precise argumentation where to ground the solution we want to present. To this aim we have to better specify the scenario we want to consider. Then we discuss the theme of the integration of the new functionality with the possibly pre-exiting technology.

2.1 More about the considered Common Working Space

We described our main goal as the construction of a Virtually Augmented Common Working Space. This locution recalls the notion of Common Information Spaces widely discussed in [4]. The two notions, although similar in some respect, show a different emphasis: the latter emphasizes information while the former emphasizes work. We believe that the emphasis on information is too narrow to capture all the aspects we are interested in. In fact, the reference metaphor and almost all the field studies used to illustrate the concept refer to an enlarged view of a repository of information. Instead, we aim at a space populated by actors, activities and all kind of resources, all considered on the equal foot. Hence, this space is dishomogeneous and distributed. Moreover, its structure is dynamic since the above entities can be strongly or loosely connected depending on the evolution of the state of the cooperation happening in the space. For us the term work is more adequate to represent the above richness. The co-related complexity makes this space sometimes difficult to be perceived by the cooperating actors. In fact, it is not identified through a single or well recognizable physical or logical space. The distributed activities make parts of the space opaque for some observer: the pertinent information is not reachable and cannot be used for coordination purposes. But we know that the awareness of what is going on, in a close or peripheral way, is the glue that makes the space exist, take a meaning for the involved actors and play a role in cooperation. In this scenario the use of a technology supporting the perception of the common working space is especially motivated since it provides coordination capabilities that are otherwise impossible [5].

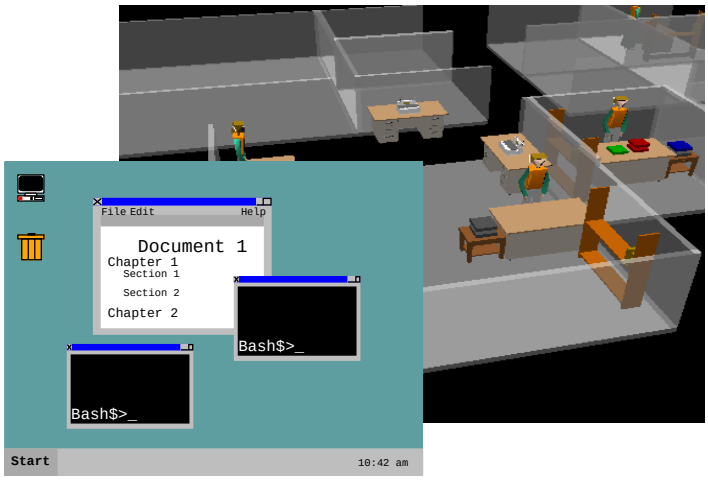

Figure 1: Representation of the focused working space and of the awareness information.

On the other hand, augmenting the capability to perceive the above complexity can overwhelm the actors and make the related information practically useless. Hence, it should be possible for each user to access exactly what is pertinent to her current interest. With the concept of *focus* we denote someone's attention for some specific aspects of the space. For example, although people are normally involved in several tasks at the same time, however nobody is executing more than one task in any given moment. Hence, people switch, more or less frequently, from one task to another and in so doing they change the point of view from which they acquire awareness from what is going on in the common working space. We want to use the notion of *focus* for two purposes: first, to create a place where actors can concentrate on their current duty or interest and find the related resources; second, as the perspective (the source of the geometrical axes) from which the broader cooperation space is looked at (represented). This choice has in turn two implications. On the one hand, the information the actor is focusing on and the awareness information have to be easily distinguishable. The way in which we want to obtain this result is to use different supports: typically, two screens devoted to contain the representation of the focused working space and of the awareness information, respectively (Figure 1).

The same idea is adopted in [6, 7] and its effectiveness has been discussed in [12] on the basis of a field investigation. On the other hand, awareness information has to be represented by taking into account different degrees of 'proximity' with the current focus: from very close to fully peripheral. This helps actors to grasp in an immediate way an important feature of awareness information. To this aim we conceive the entities populating the common working space as linked by different types of relations: from simple physical proximity up to pure logical relations. These relations define the texture of the space and allow the definition of what we call logical distance between entities. By taking the perspective of the observing actors together with their current *focus* and by combining the mutual distances between entities (a sort of transitive closure on the graph defined by the texture) we can compute the distance between the observer and each observed entity.

While physical distance can be univocally obtained by a simplification or abstraction of the real work setting, logical distance is more difficult to define since the relations constructing the texture of the logical space are, in general, not ready at hand. This is where the integration of the technology supporting cooperation with the one promoting awareness plays a fundamental role. As illustrated in [13] the needed relations can be derived from the logical

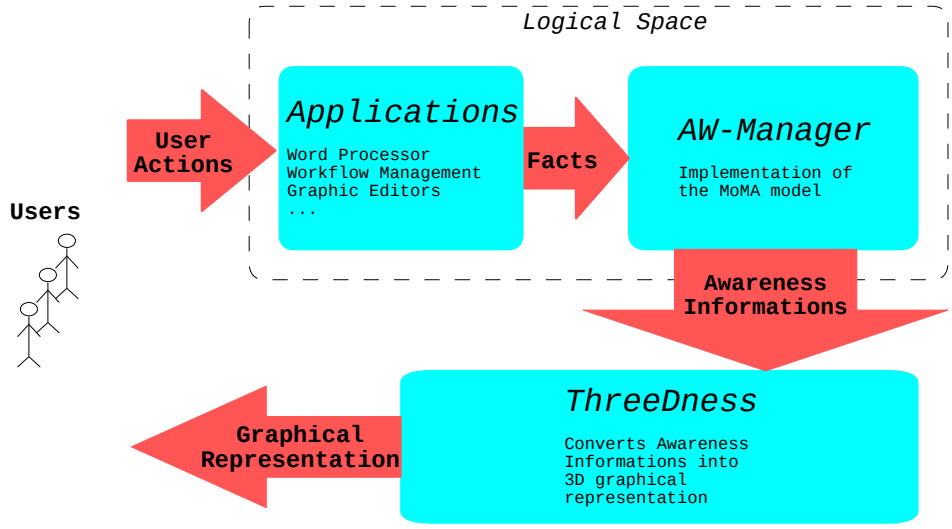

Figure 2: Information flow.

structure of the applications supporting cooperation, possibly augmented by ad hoc relations for sake of awareness. The same paper discusses how this can be done by different types of actors (playing the role of application designers and/or end-users) through the functionality provided by a software module, called *AW-Manager*. Obviously, this type of solution is not mandatory: the lack of a similar integrated functionality requires other strategies to determine the relations that can be meaningful for sake of awareness promotion. The proposed approach has the advantage to capture, 'at the source', the information related to the cooperative applications concerning articulation work as well as the field of work [14]. And more importantly, it allows for the dynamical updating of this information according to changes affecting the applications in the course of their execution.

2.2 The information flow

For the above reasons we consider the scenario illustrated in Figure 2. Users are the source and the destination of the awareness information. They play as source by using the applications supporting coordinated activities. The related *facts* are passed to the component in charge to transform them into awareness information. The latter is interpreted by the component managing its representation and finally presented on the user interfaces. The information flowing through the different components has to be codified so that the next component can elaborate it. To this purpose, the central component, in our case the *AW-Manager*, plays a pivotal role. The *facts* it is able to interpret mainly concern: the creation/deletion of entities and of the involved relations (i.e., the evolution of the space), and the change of state of the entities and of their position in the logical space (events like an actor assuming a role or activating a task or accessing a resource change the proximity of the related entities). These *facts* are transformed by the *AW-Manager* into awareness information on the basis of the awareness model it incorporates. The latter (called MoMA) belongs to the class of the Space-based Awareness models [6, 7, 8, 9, 10, 11] and is based on a reaction-diffusion metaphor. The *AW-Manager* elaborates the incoming *facts* in terms of modifications of the awareness (logical) space or of the proxies of the application entities populating the space itself. In addition, it generates the related awareness information on the basis of scripts of primitives that define the implications of the above modifications. For example, a script can establish that when an

actor activates a specific task a notification has to be generated. The target entities are determined on the basis of the texture of the space, their logical distance from the source entity, the strength of the message to be notified, the propagation law associated to the notification (for example, the original strength can decrease with the distance), and finally the threshold of sensitivity of the candidate receiving entities. Hence, the awareness information to be represented concerns the changes of states and position of the entities, and the type and strength of the notification reaching the entities that are proxy for an actor. The 'values' of the notification to other types of entities are possibly used to define and elaborate the propagation of awareness information but are not involved in any representation at the user interfaces.

The last component, that we call *ThreeDness*, is in charge of the graphical representation of the awareness information received from the *AW-Manager*. This component is illustrated in the following sections for what concerns its functionality and internal architecture.

3 The current prototype

In order to check the feasibility of the basic ideas illustrated in the previous section we developed a prototype to be used as a demonstrator. It does not contain all the features fulfilling our short and long term goals. However it serves as a test-bed for the preliminary versions of the functional and the architectural solutions. In the next section we will discuss the improvements we are working on, also in relation to this experience. In order to construct the demonstrator we have to identify a reference scenario: it has to be general enough to describe a wide range of working environments, and concrete enough to create a natural mapping between logical entities and graphical visualization. Moreover, in order to avoid, at this stage, serious integration problems with the *AW-Manager* (whose implementation is still under improvement) the demonstrator is built as a stand-alone component which receives awareness information *facts* concerning potential cooperative applications from an interface devoted to simulate this interaction.

3.1 The reference scenario

The scenario considers a work arrangement in which actors cooperatively accomplish tasks. Tasks are steps of a distributed project and require resources. This simple scenario allows us to take into account both *focus* (on the cooperation around a single task) and *various degrees of peripheral awareness* concerning other tasks as well as the related actors and resources. The scenario is described in terms of entities and relations among them as follows.

3.1.1 The texture of the logical space

The reference logical space is populated by four types of entities: *Task*, as the unit of the cooperative work; *Role*, defining the *Actor*'s responsibility for Tasks and the *Resources* used in the accomplishment of Tasks.

There are two types of resources: informative resources and tools. Tools are all kind of physical resources, as computers, printers or desks. Informative resources are of a more abstract type, as a working document.

The above entities are related with each other in the following way:

Actors <-> Roles An actor can play different roles and vice versa.

Actors <-> Resources An Actor can use more resources at the same time, but a resource can be used at most by one actor.

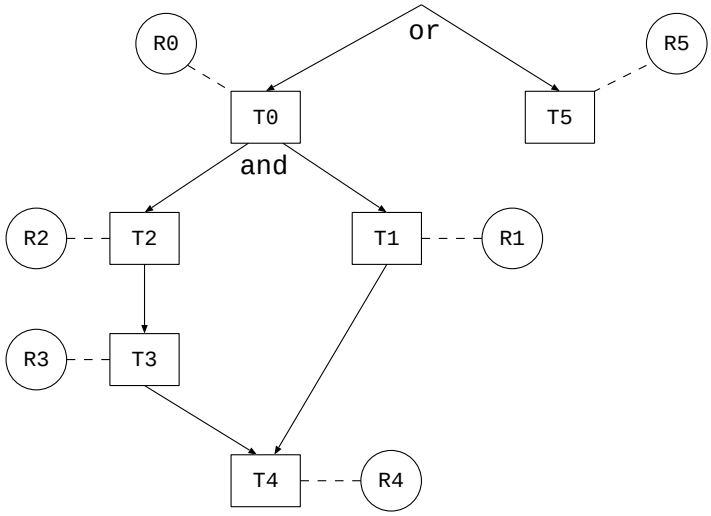

Figure 3: AND/OR graph of a task-split project.

Roles <-> Tasks A role is responsible for a task. A task can have just one responsible role which can be played by several actors. Hence, the relation between Actors and Tasks is mediated through the notion of Role.

Roles <-> Resources A role is responsible for a resource. A resource can have just one responsible role which can be played by several actors.

Tasks <-> Resources The resource is necessary for the accomplishment of a task. A task may need several resources, and a resource can be used in the accomplishment of several tasks.

Task <-> Task A project is described as a partial order among tasks to express priorities and constraints in their accomplishment. This relation is represented by an AND/OR graph. Figure 3 shows an example. T0, T1, ..., T5 are tasks. R0, R1, ..., R5 are the roles assigned to their accomplishment. To develop the project one can choose to start with task T5 or to follow the alternative branch (OR). In this case, task T0 has to be finished before tasks T1 and T2 can start: the latter can be developed in parallel (AND). Before task T4 is started, task T1 and T3 have to be concluded. The same holds between task T3 in relation to task T2.

3.2 The resulting Virtually Augmented Common Working Space

In this section we describe how *ThreeDness* visualises the above scenario. First, we illustrate the representation of the background scene. Then we describe how is it personalized and augmented for each user.

3.2.1 The visualization of the background scene

Figure 4 shows a snapshot of the three dimensional scene visualizing the background scene. The visualization is based on a simplification of the physical space.

The leading idea is to see the latter as composed of rooms, each containing the work places of the Actors living in it. A work place is composed of an avatar, a desk, a table and an archive. The avatar represents the Actor. The desk is its local working space, that is, the place the Actor focuses on while working on a specific task. The table represents the place where the Actor piles suspended tasks, that is, tasks in which it is involved but not currently working

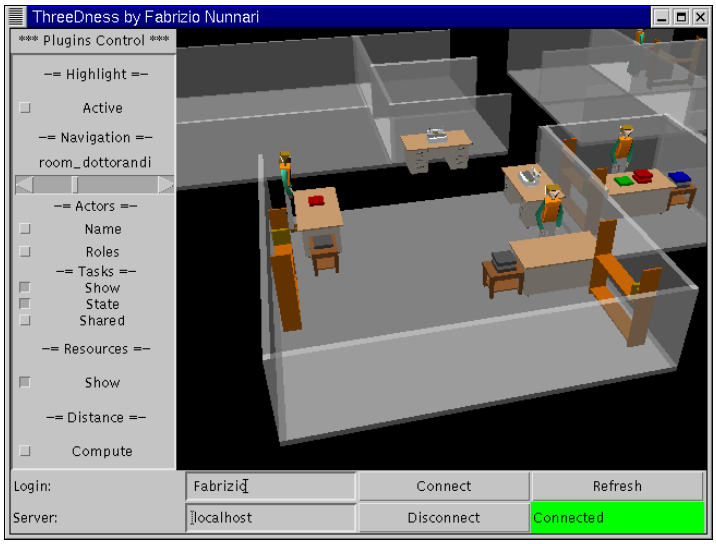

Figure 4: The background scene.

on. The archive represents the place where an Actor locates its information resources: they will be part of the personalized working place discussed in the next section. Tools can be located in common spaces, like corridors, or be part of a working place. Figure 4 shows an example of both cases. The same figure contains empty rooms. They represent a portion of the space that does not play any role in the considered cooperation arrangement. However, their presence improves the representation of the physical setting and can help users to move more easily in it.

There is a one-to-one correspondence between *ThreeDness* users and avatars in the scenario. Hence every *ThreeDness* user sees both the representation of himself and of all the other users. The first time a user activates the *ThreeDness* application, a window pops-up presenting the background scene, built of rooms and work places. The latter are used as basis for additional graphical representation of the awareness information we want to promote. At this point, each user can set some parameters through the panel (shown in Figure 4) in order to define which type of information will be added to the background. This selection can be changed at any moment, through the same panel. From now on, each user interface will be different: partly because of the above choice, partly because the background will be shown from the logical and physical perspective of each user, as illustrated in the next sections.

3.2.2 The focus *and its closer awareness information*

As discussed in Section 2, we propose to use two supports (typically, two screens) in order to represent the current *focus* and the awareness information about the surrounding Common Working Space, respectively. In this view, the first screen contains the interface of the application used to perform the focused task. Possibly, this interface contains cues representing awareness information closely related to the application itself: for example, in the case of a co-authoring system telepointers or cues representing the current state of portion of the text, and the like. The second screen contains the background scene illustrated above, enriched with additional information. In our reference scenario a key information concerns tasks: hence we describe how tasks are represented and the resources they use (Figure 5).

The Tasks an Actor is working on are represented as little coloured boxes on the desk or on the table of that Actor. Boxes are labelled with the name of the Task they represent.

Figure 5: The representation of Tasks.

Both their position and colour convey a meaning. Boxes on the table represent Tasks that are already done (grey) or that will be activated in the future (blue), according to the partial order relation described in Section 3.1. Their mutual position in the pile represents the temporal order by which they have been or will be executed. Boxes over the desk represent tasks that are currently under consideration (red). The particular Task whose specialized working environment is the content of the first screen, is still represented with a box on the desk, but with a green colour.

The representation of the Resources requires a specific comment. Recall that a Tool is a physical object while an Informative Resource may be not. Tools are represented by three dimensional objects, possibly abstracting from details, that everyone can recognize as such: e.g. a printer. Informative Resources are more difficult to represent: in fact, they can be a piece of paper, or a little book, or a text file in some computer. Beside the content it could be useful to represent the media used to keep it. Hence, such a representation is more subjective that in case of Tools. In the construction of the demonstrator we adopt a simplified solution and represent Informative Resources as folders in the archive. Again, different colours describe their state (e.g., in use or idle) and a label describes its contents. We will come back to this solution in the concluding section. The representation takes into account the two possible relations between an Actor and a Resource. If the Actor is responsible for a Tool, the latter is placed close to its working place. If the responsibility concerns Informative Resources, the latter are located in the lower part of the archive of the responsible Actor's working place. Information Resources in use of some Actor are located in the upper part of its archive, while Tools in use are logically connected to the Task where they are used: this link is visualized through a continuous line if the related option has been selected in the above mentioned panel.

The representation of the focused space is coherent with the choice to incorporate the Actor (avatar) representing the user in the background scene together with the other Actors. We believe that this double presence naturally provides a continuity between the contents of the two screens. Moreover, the abstract representation of the information concerning the *focus* (the green task) and the user's avatar define the point of view from which the space is perceived, and therefore help the user to keep the sense of orientation in the more peripheral

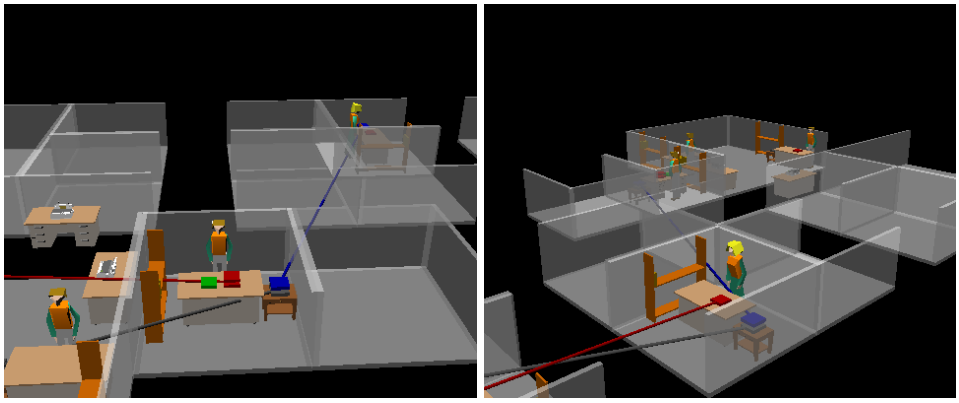

Figure 6: The Shared Tasks representation is different for different users.

awareness space: a sort of spot telling a "you are here" information.

3.2.3 Representing peripheral awareness information

All what (physically and logically) surrounds the focused Tasks can be source of awareness information. However, the latter can be of different relevance in relation to the (physically and logically) distance between the observed entity and the observer. Hence, awareness information has to be represented according to different degrees of relevance. Here below we discuss two possibilities that we considered in implementing the demonstrator.

An example of awareness information that is close to the Actor and its focused Task concerns the other Actors involved in it. Of course, this information can be part of the interface concerning the *focus* but the same information in the representation of the Common Working Space can provide additional information. Typically, this happens when some Actor is not working on this Task as the observer could expect. The representation of the Common Working Space can tell the observer something about why this is the case.

The graphical visualization of the Tasks allows the scene to contain several boxes representing the same Task, each over a different desk (or table) because each Task can involve different Actors. To make this fact evident, boxes representing the same Task are connected with a little coloured tube. Figure 6 shows an example of the visualization of the information about shared Tasks: they are connected by tubes suspended in the air, crossing walls and furniture, to highlight the fact that the two boxes at their ends stand proxy for exactly the same Task. The colour and position of the task instances together with the current *focus* of each visible Actor give the additional information we mentioned before: about which Task each Actor currently considers of higher priority (its *focus*) and the priority of the Tasks shared with the observer (their mutual position in the piles of the working place of each visible Actor). This information cannot belong to or be generated by the application executing the focused Task of the observer. The above visualization can be done for all Tasks in the working space of the observer by pointing them with the mouse. The user can choose if the visualization has to be permanent or transient (i.e., disappearing with the displacement of the mouse). Notice that, since the drawing algorithm is strongly depending on the identity of the observer, every user interface shows a different visualization of the (potentially) same awareness information.

In the above situation, the value of the distance between the connected Tasks is one, that is they are immediately connected by the (derived) relation: being an instance of the same Task. In the more general case, the logical distance defines several degrees of proximity. We choose to represent distance in terms of the dimension of the involved object. The farer is

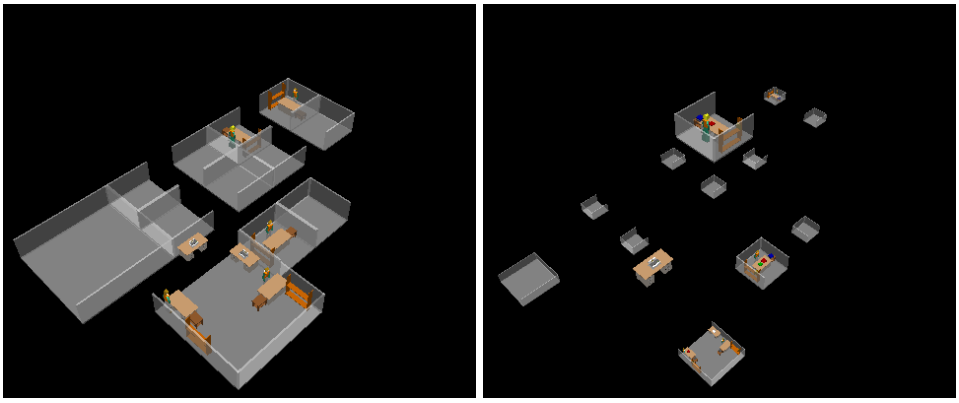

Figure 7: The Logical Distance representation is modifying the size of the rooms in the scenario.

an object, the smaller it is represented. This is coherent with the physical representation of the background scene. However, acting at the level of single object would be confusing for the observer: in fact, each room can contain different objects with different dimensions. We tried this solution and realized immediately that it was unacceptable. Hence, we decided to apply the change of size to the rooms of the scenario. For every room a resizing algorithm performs an evaluation of the logical distance between the user of the application and the content of the room. The algorithm bases the resizing on the logical distance of the elements contained in the room. A room close enough is kept in its original size, while the (logically) farer is a room, the more it is shrunk. Hence, if the content of a room is logically close to the observer, the room is big enough so that its content is clearly visible. On the contrary, if a room is logically far, it is shrunk to make its content hidden. Figure 7 shows the difference between an interface where the logical distance is visualized and another one where it does not. The picture on the left is the normal scenario, the one on the right is showing the same scenario modified by the resizing algorithm. The visualization is changing from an observer to another one. First, every user can choose through the mentioned panel whether to activate the logical distance representation or not. Secondly, the computation of the logical distance of the rooms is different for each observer because it is dependent on each user current focus. So the topology of the background scene is the same, but the size of the rooms can be different. Notice that if a shrunk room contains a task shared with the observer, the previously described tubes enter the room without any additional precision. Again, by pointing the room it can be zoomed and the more detailed information will appear.

4 The ThreeDness architecture

We describe now the internal architecture of *ThreeDness* in the general context described in Figure 2. As for the technologies used for its implementation, the code is written in Java, the network communications are RMI based and the 3D graphics engine is based on the standard Java3D extension.

In this section we first describe the client/server architecture of *ThreeDness*. Then, we give a deeper look at the *Plugin system*, a general purpose software architecture that has been incorporated in *ThreeDness*.

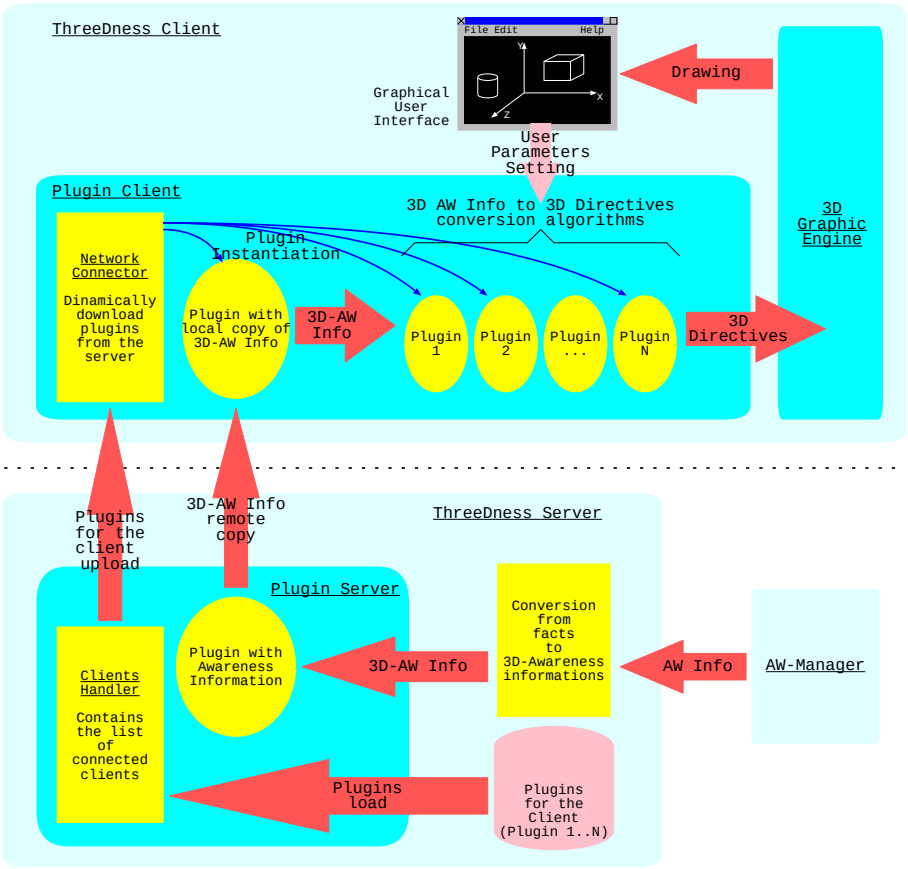

Figure 8: Client/Server Architecture of *ThreeDness*.

4.1 The Server and the Clients in ThreeDness

ThreeDness is composed of two distinct modules: the server and the client, as shown in Figure 8. The bottom part shows the server side architecture, while the top part shows the client side. The dotted line in between represents the network communication between the two sides. Following the standard client/server paradigm, there is a single machine running the server application, while several clients establish a network connection with the server.

Server side (bottom): During its execution the server communicates with the *AW-Manager*, the application in charge of collecting *facts* about users activities and transforms them into *Awareness Information*. A deeper discussion about the *AW-Manager* and its underlying architecture can be found in [13].

The *ThreeDness* server is mainly composed of three parts: A *PluginServer*, an *Awareness Information* to *3D-Awareness Information conversion* module and a *Plugins repository*. The conversion module contains a set of algorithms to transform the *Awareness Information*, coming from the *AW-Manager*, into a data structures suitable to deal with its graphical representation. In fact, the *AW-Manager* is a source of data fully unaware of any graphical representation. We call the converted information *3D-Awareness Information*. The *PluginServer* functionalities and the *Plugin system* managing them will be discussed in Section 4.2. The *PluginServer* contains a Plugin dedicated to collect *3D-Awareness Information* from the conversion module and send the related updated data to all the connected clients.

Finally, the *Plugins repository* contains the Plugins devoted to the representation of the

3D-Awareness information: they are loaded by the *PluginServer* and sent to the clients when they connect. Moreover, when they are updated for some reason, the *PluginServer* reloads them and all the connected clients are forced to download a copy of the new Plugins.

Client side (top): The *ThreeDness* client is composed of three parts: The *PluginContainer*, the *3D graphic engine* and the *graphical user interface*. As before, the details of the *PluginClient* will be discussed in Section 4.2. When the *ThreeDness* client establishes a connection with the server, the *PluginClient* downloads from the server all the needed Plugins. They constitute the main part of the code running on the client. One of these Plugins, named "Local copy of awareness informations" is dedicated to manage the copy of the *3D-Awareness Information* received from the server. The latter is elaborated by the other Plugins that are responsible to convert them into *3D Directives* for the *3D Graphic Engine* that generates the graphics for the graphical user interface.

As anticipated in Section 3.2.1, users can set some parameters to customize their 3D interface. These parameters affect the above mentioned Plugins. Notice that customization is local to every client, that is, every client receives the same *3D-Awareness Information* but visualizes it differently depending on the users preferences.

4.2 The Plugins System

The *Plugin system* is a peculiar characteristic of *ThreeDness*. First we give the basic motivations and describe how it fits into the *ThreeDness* architecture. Then we give some technical detail.

4.2.1 Plugins usage in ThreeDness

The idea of having remotely downloadable Plugins came out from two needs: the customization of the visual representation and the efficient management of clients updates. Since the combination of these two needs, namely customization and efficiency, is typical of many applications, we developed a general purpose *Plugins system* that can be used irrespective of the type of functionality provided by the managed Plugins. Of course, there are ready-at-hand solutions (like, e.g., Java beans [17]) but generally they show two problems: they are too complex to manage and more importantly they cannot be used as general RMI objects. In order to fulfill the need of customization in our framework, it is necessary to provide every client with a copy of the algorithm translating *3D-Awareness Information* into *3D Directives* for the *3D Graphic Engine*. The choice to let the server compute the 3D scenario and then distribute it to the clients has two disadvantages: although simpler, this architecture makes customization more difficult and over-charges the server with heavy and complex 3D calculations. The need of customization implies also a modular architecture that facilitates the modification of partial functionalities and their integration in the overall application. All the above considerations motivate the choice to organize all the exchanges of code and data structure between the server and the clients in terms of Plugins.

In this way, we obtain at the same time: modularity, mobility and an efficient management of the 3D computations. In fact the *ThreeDness* server does not need to deal with 3D graphics and to have any 3D library installed. This brings another advantage. With most of the code downloaded dynamically at boot time, the need of reinstalling a client on every machine, when some code change occurs, is significantly reduced. By the way, this makes easier the

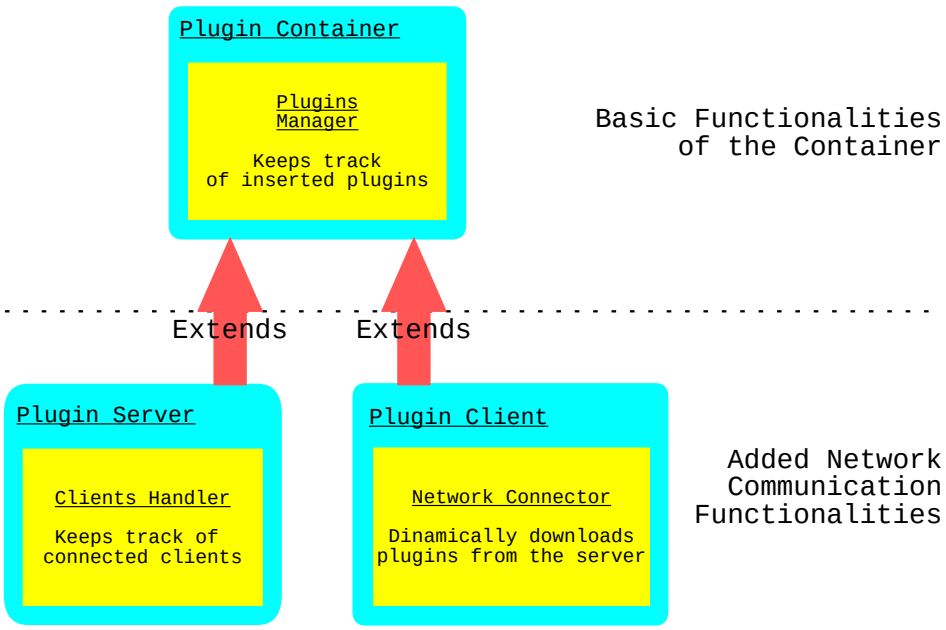

Figure 9: The hierarchy of the *PluginContainers* class.

incremental development of *ThreeDness*, where changes are quite frequent. Moreover, all clients do not need to share the same 3D libraries.

One may argue that a complete upload of all Plugins may be heavy from the point of view of the network load. Currently the application has been developed for local area networks, where transfer speed is fast enough and does not create any problem. Anyway, we believe that, if band-with becomes a problem in some situation, a cache mechanism might be implemented to reduce the data transfer by preventing the client from the upload ofPlugins that are not changed in the server repository.

4.2.2 Components and features

The purpose of the *Plugin system* is to give support for a flexible and easy exchange of Plugins. Figure 9 shows the three most significant classes of the system. The *PluginContainer* at the top of Figure 9 is a class providing the basic functionalities that are common to both the *PluginClient* and the *PluginServer* subclasses (bottom of the figure). These functionalities handle the collection of Plugins in order to:

- Insert and remove Plugins to and from the container. This is the basic feature that allows the designer to decide which Plugin are available in *ThreeDness*.
- Initialize and destroy a Plugin once it's inserted in the container. Every Plugin has the responsibility to initialize itself after being inserted in the container and to de-initialize itself before being removed.
- Realize Plugins inter-communication. With this mechanism every Plugin has the list of all the other Plugins inserted in the container and can interact with them.

The two sub-classes derived from this class add networking functionalities to the *Plugin-Container*. Normally, when instantiated, a *PluginContainer* is empty. The *PluginClient* is a special case of a *PluginContainer* with the additional capabilities to dynamically download

Plugins from the specified server. The *PluginServer* is a specialized *PluginContainer* with the additional capability to upload Plugins to a client requesting them. Note that the Plugins contained in the server are different from the Plugins that will be sent to the client. In fact the server is containing two sets of Plugins: one set for itself, as extension of a stand-alone container, another one of Plugins that will be uploaded to the clients. In addition, the *Plugin system* has the possibility to force all clients, at run-time, to de-initialize and remove the contained Plugins and to reload a new set from the server.

5 Conclusions

The paper presented an approach and an architecture supporting the representation of awareness information in cooperative applications. The approach is based on the choice to make a clear distinction between the representation of the focus and of the various degrees of awareness information. The architecture is based on the choice to support flexibility and adaptability of the representation according to users needs. The presented demonstrator served as a test-bed for the initial solutions as well as a means to collect initial impressions from people with different interests: potential users and interaction designers. Irrespective of the evident limits of the demonstrator, *ThreeDness* was perceived as an original approach to the problem. By the way, it served as a vehicle bringing the themes concerning the promotion of awareness information to professionals involved in single-user interfaces design: they are almost unaware of the complexity inherent to cooperative applications. They recognized it as a quite challenging field of research where consolidated techniques are of little help. Moreover, they pointed to some application areas (typically, remote process control) where similar ideas could be profitably applied. Perhaps, an interesting outcome was that they recognized that also the design of single-user applications could profit from our approach.

The new version of *ThreeDness* is based on the architecture presented in Section 4. The flexibility it provides is currently at the infrastructural level as a fundamental step toward flexibility at the application level, where we aim at the highest degree of customization. This means, customization is not only based on tuning parameters but more substantially on the choice of the appropriate metaphor to represent the background scene as well as the personalized awareness information. The choice to let the representation be constructed at the client side, under user control, realizes this approach without creating interpretation problems. In fact, the basic information is the same at each site, the only difference being how it is filtered and represented. When information goes from a site to another, it is represented here according to the local customization. Moreover, the virtual space is used for sake of awareness promotion and not necessarily as an arena where to perform cooperative activities.

Finally, the distinction between focus and awareness space seems to open the possibility to apply the proposed approach in the case in which the focus interface is incorporated in objects distributed in the physical space. In this case, the awareness information represented on the 'second' screen can help to reconstruct an even more distributed space whose perception depends on both the logical and physical position of the current focus of the observer.

References

[1] K. Schmidt, C. Simone: "Mind the gap! Towards a unified view of CSCW". In COOP2000. The Fourth International Conference on the Design of Cooperative Systems, Sophia Antipolis, France, 23-26 May 2000, 2000, pp. 205-221.

[2] S. Donatelli, M. Sarini, C. Simone: “Toward a Contextual Information Service supporting adaptability and awareness promotion in CSCW systems”. 2000. Designing Cooperative Systems. R. Dieng et al. IOS Press. pp. 83-98.

[3] A. Drozd, J. Bowers, S. Benford, C. Greenhalgh, M. Fraser: “Collaboratively improvising magic: An approach to managing participation in an on-line drama”. 2001. Proceedings of the 7th European Conference on Computer-Supported Cooperative Work. pp. 159-178. Kluwer Academic Publisher.

[4] L. Bannon, S. Bodker: “Constructing Common Information Spaces”. 1997. Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work, 12-16 September 1999, Copenhagen, Denmark. Kluwer Academic Publisher. pp. 81-96

[5] M. Büscher, P. Mogensen, D. Shapiro: “Spaces of Practice”. 2001. Proceedings of the 7th European Conference on Computer-Supported Cooperative Work. pp. 139-158. Kluwer Academic Publisher.

[6] W. Prinz: “NESSIE: An Awareness Environment for Cooperative Settings”. 1999. Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work, 12-16 September 1999, Copenhagen, Denmark. Kluwer Academic Publisher.

[7] V.A.: “Tower: Cooperation Awareness in a Populated Information Landscape”. 2001. Seventh European Conference on Computer-Supported Cooperative Work, Conference Supplement-Demonstrators, pp. 99-101.

[8] C. Greenhalgh, S. Benford: “MASSIVE: a collaborative VE for Tele-conferencing”. 1995. ACM TOCHI, vol. 2, no. 3, pp. 239-261. “http://www.crg.cs.nott.ac.uk/research/systems/MASSIVE”.

[9] S. Benford, C. Greenhalgh: “Introducing third party objects into the Spatial Model of Awareness”. 1997. In 5th European Conference on CSCW, Lancaster - UK, ed. W. Prinz, T. Rodden, and K. Schmidt. Kluwer Academic Publishers, pp. 189-204.

[10] T. Rodden: “Populating the Application: A Model of Awareness for Cooperative Applications”. 1996. In CSCW ’96. Proceedings of the Conference on Computer-Supported Cooperative Work, Boston, MA, November 16 -20, 1994. New York: ACM Press, pp. 87-96.

[11] O. Sandor, C. Bodgan, J. Bowers: “Aether: an awareness engine for CSCW”. 1997. In 5th European Conference on CSCW, Lancaster, UK, ed. W. Prinz, T. Rodden, and K. Schmidt. Kluwer Academi Publishers, pp. 221-236.

[12] J. Grudin: “Partitioning Digital Worlds: Focal and peripheral awareness in multiple monitor use”. 2001. Proceedings of the CHI 2001 Conference. March, 2001.

[13] C. Simone, S. Bandini: “Integrating awareness in cooperative applications through the reaction-diffusion metaphor”. Computer Supported Cooperative Work: The Journal of Collaborative Computing - Special Issue on Awareness in CSCW (to appear).

[14] K. Schmidt, C. Simone: “Coordination Mechanisms: Towards a Conceptual Foundation of CSCW System Design”. 1996. Computer Supported Cooperative Work: The Journal of Collaborative Computing, Kluwer Academic Publishers, pp. 155-200.

[15] M. Divitini, C. Simone: “Supporting different dimensions of adaptability in workflow modeling”. 2000. Computer Supported Cooperative Work: The Journal of Collaborative Computing -Special Issue on ’Adaptive Worflow Systems’, vol. 9, no. 3-4, pp. 365-397.

[16] S. Benford, C. Greenhalg, T. Rodden, J. Pycock: “Collaborative Virtual Environments”. 2001. Communications of the ACM, vol. 4, no. 7, pp. 79-85.

[17] http://java.sun.com/products/javabeans/