**Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση**

Τόμ. 1 (2000)

2ο Συνέδριο ΕΤΠΕ «Οι ΤΠΕ στην Εκπαίδευση»


# Collaborative training utilising audio-visual, multilingual, and interface agents' enabled Virtual Community Servers.

*Constantinos Davarakis, George Koutalieris*

**Βιβλιογραφική αναφορά:**

Davarakis, C., & Koutalieris, G. (2025). Collaborative training utilising audio-visual, multilingual, and interface agents' enabled Virtual Community Servers. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 701–710. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/8309

# Collaborative training utilising audio-visual, multilingual, and interface agents' enabled Virtual Community Servers.

***Dr. Constantinos Davarakis,***
*Managing Director,* costas@systema.gr,
***George Koutalieris,***
*Information Systems Manager,* gkout@systema.gr,
Systema Informatics SA,
215 Messogion Av, 115 25 Athens, Greece
Tel: +301-6743243, Fax: +301-6755649

**Key words:** Groupware, Community-ware, Autonomy agents, avatars

## Περίληψη
Στοχεύοντας στην ανάπτυξη ενός ολοκληρωμένου περιβάλλοντος υποστήριξης της εκπαίδευσης, μέσα από την εκπροσώπηση όλων των παραγόντων και συντελεστών μίας εκπαιδευτικής διαδικασίας εντός ενός εικονικού περιβάλλοντος, η παρούσα εργασία σχεδιάζει και αναπτύσσει μεθόδους υλοποίησης που θα επιτρέπουν «σύγχρονη εκπαίδευση» ικανή να υποστηρίξει όλες τις γνωστές παιδαγωγικές προσεγγίσεις συνεργατικής διδακτικής που μπορούν να εκμεταλλευθούν τις νέες τεχνολογίες. Το έργο χρηματοδοτείται εν μέρει από το 5ο Πλαίσιο για την Έρευνα και την Τεχνολογική Ανάπτυξη της Ευρωπαϊκής Ένωσης και το Πρόγραμμα «INVITE» των Τεχνολογιών Εκπαίδευσης της Κοινωνίας της Πληροφορίας.

## Abstract
Aiming at the development of an integrated learning environment; which will enable the realistic representation of all factors and agents that take part in an education process within a Virtual Environment; this work designs and gives implementation guidelines that will activate "synchronous tele-learning" building especially in the collaborative aspects. This work is partly funded by the project "INVITE", under the 5th RTD Framework's IST Programme.

## 1. Introduction in Collaborative synchronous training
Differently from other animals that are able to live separately in reasonable manner, humans tend to organise themselves into societies in order to survive. Since remote human origins man, has survived and made progress because he has lived in groups and learned to divide tasks. By dividing tasks and using supporting tools, quality and productivity can be improved, and human beings empowered through collaboration. Today, living in the information era, people can perform several activities without regards to geographical location. This has become possible because of the big merge between the computer and communication industries. In that sense it is possible for people to interact with colleagues, share data and computational resources or access information in digital libraries. For instance, research colleagues can work collaboratively in a Physics Collaboratory as reported by Agarwal et al [1]. Another example is that of a collaboratory based on virtual records which allows health practitioners to work together more effectively as reported by Kilman and Forslund [2]. The Digital Agora by Watters et al [3] is a Web-based collaborating system where the aim is to provide support for active learning in Social Sciences. Other applications that can be considered are Digital Libraries [4], and Web-based Entertainment like multi-user Web games. Web browsing can also be viewed in a collaborating setting as stated by H. Lieberman et al [5].
A special information technology has been developed in order to support and facilitate the work of groups of people. This technology is called groupware and can be used to communicate, cooperate, coordinate, solve problems, compete or negotiate. Traditional technologies as telephony often

qualify as groupware, however the term is ordinarily used to refer to a specific class of technologies relying on modern computer networks.
Groupware is typically categorized along two primary directions:

1. whether users of the groupware are working together at the same time ("real-time" or "synchronous" groupware) or different times ("asynchronous" groupware), and
2. whether users are working together in the same place ("collocated" or "face-to-face") or in different places ("non-collocated" or "distance").

The main aim of the system presented in this paper is to build a platform for synchronous telelearning, which can be interfaced with standardised content management and instructional management systems. New technologies in distributed Virtual Reality are utilised and their application for new methods in education is being investigated. A real-time educational environment, where presence and attendance to students will be made compulsory for inscribed students with access to the Internet, will be implemented for that matter.
Today there is a growing trend towards the development of software that enhances the formation, maintenance and support of digital communities; we usually refer to this type of software as communityware [5].
Key factor contributed to this trend, is the Web explosion that has led to the development of digital communities around certain web sites, chat rooms, Usenet groups and even multi user virtual environments (Multi User Dungeons, MUDs) that reside on the Internet.
Another contributing factor to the development of communityware is the simultaneous growing trend in intelligent agent applications. This type of software provides proactive assistance in using highly interactive software [8] [9]. Interface agents in particular, emphasize autonomy and learning in order to perform tasks for their owners [7]. The key metaphor underlying interface agents is that of a personal assistant who is collaborating with the user in the same environment [10]. Research in fusing agent-based technologies with groupware is also motivated by the fact that the growing complexity computer environments is beginning to outgrow simple direct manipulation interfaces.
Agent software uses techniques from artificial intelligence such as machine learning, knowledge representation and inference, and may also employ heuristic statistical techniques. Agents are often cast in a role of making suggestions to a user rather than taking definitive action. Most agents are in the service of only a single user and deal only with the interests of a single user.
Using special groupware, in the context of learning and training, can produce interesting results. Collaborative training - whether via instructional software on a PC or over the Internet - is slowly gaining momentum among developers and end-users of training and educational software. A key motivation is that this type of software lets training be a continual process rather than a static presentation of learning material. Analysts and users state that for some students; enterprise application training on computers is inherently better than a live classroom. Certain products for example, let users send private e-mail messages to the instructor while the training is taking place. That is a real advantage for learners who aren't comfortable admitting in front of other people when they don't understand something. Using certain groupware makes it easier for instructors to control the virtual classroom, ensuring that overeager participants don't dominate the proceedings [11].
There is a growing need for groupware that supports distributed cognition, namely the notion of people thinking and solving problems collectively. This type of software should focus in how people communicate and jointly use artifacts to accomplish joint work.
In following, prototype technological specifications determining a groupware for the support of distributed cognition in a distributed synchronous collaborating environment for training, is presented. The anticipated system will use interface agents to assist users through the training process and perform knowledge management for the system. The implementation will be based on

certain virtual communities that will form the population of highly interactive 3D virtual environments.

## 2. Technology profile

In this section the technology features that could be made available for the design of teamwork oriented learning environment, containing synchronous and asynchronous training approaches, are given a closer look (see System Architecture in figure 1). In following the described technology suite enables the expansion of a whole range of possibilities for innovative ideas in education and training.

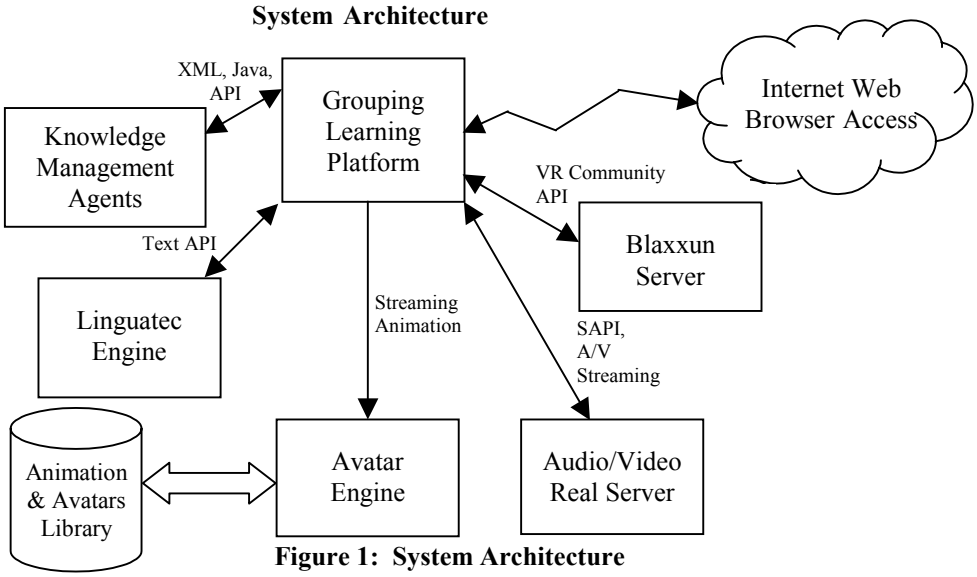


**Figure 1: System Architecture**

### 2.1 3D Multi-user Environment

The term 3D Multi-user environment is characterised by the following parameters:

- based on 3D Environments (VRML) with smooth movements (seamless)
- extensive textures
- 3D-Icons for manipulation of the environment
- stereoscopic visualisation options

In this work the problem of involving VR technology with audiovisual applications, multilingual capabilities and interface agents support is dealt through the integration of commercially available solutions. Namely blaxxun Community Servers, Audio/video streaming engines, Linguatec translation engine and Knowledge Management Servers.

### 2.2 VR Community blaxxun Servers

The blaxxun Community Platform 4 is a client-server system comprised of the blaxxun Community Server Modules, the clients' blaxxun Contact and blaxxun 3D, as well as the blaxxun Community Platform Software Development Kit and Administration Interfaces.

The blaxxun Community Platform offers community features as 2D or 3D place management, including 2D/3D-avatars and agents, shared events and objects as member homes with furniture or a ball in a game. The system provides a member directory, including friend and buddy lists, real-time text chat, text-to-speech, instant messaging, message boards and calendars. Members can play specified member roles, join interest clubs and take advantage of incentive programs. The

community management allows highly differentiated member and guest management, registration and authentication by highest safety standards as well as voting and decision-making.
Blaxxun interactive supports all relevant standards: HTML, VRML, Java, ActiveX, OCX, Direct3D, and OpenGL. Other evolving standards will be supported as they emerge. The Community Platform is available for Windows NT, Linux, and on various commercial Unix flavours, such as Solaris. The client runs on Windows 95, Windows 98, and Windows NT. The 2D and communication features, as well as blaxxun3D, are also available on Mac and Unix. The blaxxun Community Platform operates with all browser versions from Netscape and Microsoft and several 3D engines (Direct3D and OpenGL – formats VRML and Superscape). 3D support is available for browser versions 3.x and later. Members behind a firewall can specify usage of a firewall-compatible protocol. A key feature of the blaxxun Community Platform is its uniquely flexible and extensible architecture. Capable of supporting thousands of users simultaneously, the server can be distributed across multiple machines and is optimised for large groups. Through its SDK (Software Development Kit), the Community Server can be integrated with existing systems, including databases, advertising management systems, billing systems, E-commerce systems, company Intranets and other backend systems. Additional modules can be hooked up with the existing services. Administration is done by an easy to use, HTML based administrative interface.

### 2.3 Augmented reality

The aim is to integrate seamlessly 2D video with 3D worlds. Furthermore the use of Photo realistic Avatars is foreseen. The chosen characteristics include cost effective implementation for Video Avatars, using lip synchronisation and employing video streams.
The AvatarMe© engine is closely looked at, since it conforms to the IEEE H.ANIM standard.
In particular, the following set of technological parameters have been specified:

***A) Audio/video streaming of available content***

- Integration of available asynchronous learning systems

***B) Voice over IP Multi-User Communication***

- Voice chat, on line Lectures, Many to many chats

***C) On line Translation***

- Online text translation to voice, Multilingual protocols

***D) Document repository***

- Data visualisation, Structured search engines

***E) Expert systems***

- Intelligent knowledge based agents

***F) MPEG-7 search***

- Video database search (e.g. content of events and lectures)

***G) Mobile protocols***

- Connected to PDA, Mobile phone, Portable communicator platform, Home TV

## 3. The application framework specifications

### 3.1 Learning Platform wish list (technology functionality)

Our requirements for the project would include a three-dimensional view of the world, a text chat with "whisper", "telepathy", and "mute" functionality, and the ability to leave email messages for users who aren't around. Special interfaces to express emotions and gestures, walking animations, and the ability to teleport (by using portals, or by "beaming" to another user's location, or by setting "spacemarks" that we can return to at later time). A user list, a thematic list, and a map are

capabilities that the Learning Platform could endorse. Eventually voice chat, video chat, stereoscopic rendering and many other features could be added to the list.

### 3.2 Basic System Architecture

In order to support all this, and allow easy upgrading in the future, the Learning Platform architecture needs to use a highly modular design. We see four basic modules on the client side of the system, which need to be visible at the prototype phase: chat +, user interface and 3D view, profiling &intelligence (with Memory), networking.

### 3.3 The Chat+ Module

The chat+ module needs to be designed to be both a source and a sink for chat information. We should be able to accept typed input from the user (along with an indication of whether the user is speaking, whispering or sending a “telepathic” message). We should also be able to display multilingual text to the user, with an indication of who sent it and whether it’s a whisper or a regular message. We also want to be able to display system messages in the chat window in order to keep the user informed of changing conditions ("System going down", for example).

The reason for making the chat+ subsystem a separate module is that it allows us to easily upgrade it later. For example, text chat may not be appropriate in a fully immersive environment such as a CAVE or an HMD, and perhaps multiple languages could not be supported in synchrony at this stage. In that case, later we could replace the chat+ module with one that handles voice chat. All the functionality of the existing chat module would be preserved - for example, incoming text chat messages would be put through a text-to-speech converter. By keeping the design modular, we can introduce features such as voice chat (and even "video chat”) more easily later on.

### 3.4 The View and User Interface Modules

The 3D view module is what gives the user their sense of "presence" in the world. It allows them to see the virtual environment, and interact with objects in it. The interface is inherently two-way, since users want to be able to view the world and also interact with it, and navigate through it.

Conceptually we wouldn’t want to lock ourselves into a particular viewing metaphor. For example, we want to be able to support a window-on- the-world approach, with an on-screen window giving a simple view of the world which the user can interact with by clicking their mouse. We also want to support more esoteric, viewing systems such as head-mounted displays, projection devices, and more. We also want to be able to support a range of input devices such as space-balls, gloves, wands and so on. Regardless of what the actual viewing/interaction technologies are, we want the software interface to the 3D view module to remain the same.

The UI module is what lets the user control their experience of the virtual world. It lets them perform tasks such as setting space-marks, beaming to other locations, creating new objects, leaving the world, etc. Especially for use by the trainers, we need to cater for Human Comfort-ability. Once again, we want the design to be modular. Our initial implementation for the prototype, will probably use a fairly traditional GUI (graphical user interface), but we want to keep our options open. In particular, we want to support a three-dimensional user interface that’s part of the world, which would make more sense for an immersive VR experience. We also want to support speech input, for those cases where even a 3D UI would be too limiting.

### 3.5 Profiling, Intelligence and Memory

To personalize the access, the requirements should include: persistent identities (complete with profiles and custom avatars), as well as the ability for users to create new training objects with persistent state. Users should be able to build homes for themselves, and create "bots" or agents. Users should be able to own objects, and copy or "sell" them (which implies a form of electronic commerce). Message boards and newspapers are also important features for creating a sense of community. A necessary prerequisite for attaining intelligence to support both synchronous and

asynchronous training modes, is being well informed about the learning methodologies and the educational content. Hence there should be the means to manage and maintain the insight and background knowledge participants have gained during the course of the learning sessions.
To achieve this goal [7, 14]:
(1) Users need to be provided with specialised, easily configurable search engines and a means to organise and manage their findings.
(2) Specialised research on particular topics will be facilitated: users can train agents to search for information according to their personal perspective on a topic. These agents, could be shared and reused by other users, thus the knowledge contained in the training strategy can be shared (both the content, which the agents carry, as well as its search strategy). Also, agents can be trained by a group of users collaboratively.
(3) A responsive, rewarding and self explanatory interface will make it enjoyable for new users to actively participate: users will not have to read manuals and start from "scratch", instead, they will be able to build on the work of others and join structures built by more experienced users and start from there. Also, users will be able to distribute their roles between more administrative or organisational tasks, and more research related tasks.
These tools will provide additional incentives and value to users, by helping them to quickly gather information about a certain topic, sustain further research and deepen their knowledge.
Ultimately the Knowledge Management System (KMS) will provide conceptual specifications for such functions and also a conceptual model for the structuring of the content to be provided for a first orientation on certain topics [7]. The **Learning Platform** would then not only provide means for temporary sessions, but would also have a lasting benefit for the users, and might ideally generate new online communities for certain topics. Moreover, the KMS component will provide tools for personalised searching and facilitate the organisation of background information [15].

### 3.6 The Networking Module

In many ways, this is the key to the whole system. The networking module is what allows communication between entities in the world, including the text chat and avatar movement as well as state changes. The networking module also needs to provide a way for users to create and destroy objects, modify them, and so on.
Our initial design goals include support for a range of networking technologies and topologies. In effect, the network module is a wrapper around any of a number of different underlying systems. For example, the alternatives are either to choose to deploy own low-level networking software, or to leverage existing technologies such as IRC, MUDs, and so on [12]. Ultimately, we might have a number of different network modules which can simply be dropped into the system depending on the specific needs of the application.

### 3.7 Implementation

We need to have the initial outline of our design ready to start talking about implementation. One of our design goals [13] was to create a system that is as cross-platform as possible. The obvious language choice, especially for an internet-based project, is Java. Once we've decided to use Java, a number of other choices become clear: The View module will probably be implemented using Java3D in relation to the Blaxxun3D applet, and we could even investigate compatibility with other environments (e.g. Shout3D). Java3D supports stereoscopic rendering, and has a viewing model that was designed for head mounted displays and CAVE systems. Even though those features won't be in our initial implementation, it's nice to know that there's a way to do it later on.
One implementation of the Networking module will likely use the Java Shared Data API. The Avatar management will need to utilise the Blaxxun API Server, and the Agent Server triggering gestures to the AvatarMe Avatar Engine. The UI front end can be built using successful

composition technologies for 3D environments (to be decided), and the chat will utilise the TextAPI. Since we want users to be able to easily create their own avatars and other objects, we need to choose a graphics file format that is widely supported. Despite the unanimous functional problems, VRML is still the best choice. All the major 3D authoring tools support direct export to VRML, and other formats can easily be converted into it. The Java3D scene graph is based in large part on VRML, and there are off-the-shelf VRML import libraries for Java3D. Features such as voice chat and video chat can be handled by the Java Media Framework (JMF), and the resulting audio can be spatialized. Finally, depending on the user scenario the KMS; with or without Content brokerage and document management support; could be developed using Autonomy Knowledge Server.

### 3.7.1 Network Implementation

The details of the networking module, which is by far the most complex and intricate part of the system is further analysed. The entire architecture of the system needs to be outlined before we can discuss the specifics of the networking module.

The multi-user system will be able to use a variety of underlying networking technologies. A MUD server, an IRC server, or JSDA as our substrate, and from the application standpoint it should all be completely transparent.

Regardless of the technology used, the networking module needs to provide two basic functions. It needs to provide a way of exchanging data (text chat, position updates and so on) between users, and it needs to maintain consistent and persistent state information for all the objects in the system.

### 3.7.2 Observers and Participants

One feature that will set our system apart from most the others [12], that are out there is that we will draw a distinction between "observers" and "participants". This will be useful for a range of applications. For example, we may decide to have a virtual theatre performance, in which most of the people attending the event will be passive observers and only a few (the actors) will be active participants. Another example would be a virtual object that's driven only by real-world data (such as a stock market feed), and which doesn't need to observe the virtual world even though it's participating in it. A human-controlled avatar combines both observer and participant. The observer part is responsible for showing the user the virtual world, and the participant part is responsible for letting the user control their avatar and interact with the world and its contents.

### 3.7.3 Locations

One of the challenges faced by multi-user worlds is scalability. The amount of data generated by a large number of users is not only a problem from a bandwidth standpoint, it also puts a burden on the client-side rendering engine, as well as the user's ability to follow what's going on. This gets even more difficult when we begin to consider voice chat and video chat.

To help manage the data, we'll use the same approach adopted by most existing multi-user systems. Instead of treating the world as a single large space, we'll subdivide it into a number of distinct "locations", similar to rooms in a house. Things happening in one location will not necessarily be relevant to users in other locations, so we can do a kind of "relevance filtering" to limit the amount of data that must be sent. Users in the same location can converse and interact with each other, without having to deal with information about what's going on in other parts of the world.

In our initial implementation, we'll keep things simple and only allow an observer to be in a single location at any given time. A more sophisticated implementation would allow the user to be in multiple locations simultaneously. If they're in the living room, they wouldn't just care about things going on in the living room - they might also want updates from the adjacent kitchen, dining room and front hall (but not from the garage or the backyard). A location is basically an abstraction of the filtering mechanisms of the underlying networking technology. In other words, a location would

correspond to a “room” in a MUD environment, a channel in IRC, or a JSDA channel. All of those constructs serve the same purpose - they define a context for interaction.

### 3.7.4 Entities

A set of “entities” exists within each location. In our system, an “entity” is anything that has dynamically changeable state. This includes not only avatars and bots, but also such simple objects as a flashlight that can be on or off or a whiteboard that users can scribble on.

Each entity has certain basic properties. In addition to having a unique identifier, an entity has a textual description and the URL of a file containing a visual description of the entity. There will also be information about when the entity was created, who owns it, and so forth.

There are a number of subclasses derived from the base “entity” class, such as “avatar” and “prop”. There is also a “location” class, and there will be a location entity associated with each location in our world. It is these location entities that store all the basic properties of the location.

It’s important to understand the difference between a location and its corresponding entity. The location itself just provides a medium of communication, and it doesn’t have any actual properties. The entity that corresponds to that location will store a visual description of the location, a textual description, and various other properties.

Among these basic properties of a location entity is a list of “passages” to other locations. When the user goes through the passage, they are transported to the location which is the target of that passage. In sophisticated implementations, a location will also contain a list of other locations that are visible from the current one. This will be similar, but not necessarily identical, to the list of passages. For example, we may have a window that allows us to see the front lawn, but we cannot pass through the window.

In addition to the location entities, there is also a “world” entity that has information such as the name of the world, the local time of day in the world (each virtual environment can have its own timezone), a list of all the locations in the world, a list of all the currently active entities, and so on.

### 3.7.5 Pilots and Drones

Each entity in the system actually consists of two parts. We’ll borrow the terminology popularized by the Living Worlds effort, and refer to these two parts as “pilot” and “drone”.

The pilot is the “brain” of the entity, and it’s the part that reacts to stimuli, makes decisions, and maintains consistent, persistent state information. The pilot for an entity exists in exactly one place, usually as a process (or thread) running on some host connected to the system. It can maintain its persistent state information however it chooses, either by writing it to a file or maintaining it in a database.

The drone is the “body” of the entity, and there will be one instance of an entity’s drone on each observer in the system. If an observer enters a world with five entities, there will be five drones created on that observer’s client. If a new pilot enters the world, there will be a drone created for it on each observer’s client.

Note that only participants will be running pilots on their local client systems, and only observers will have drones for other participants. The networking module’s primary ongoing responsibility is to provide a bi-directional communications mechanism between the pilots and their drones. Pilots send messages to their drones over the network, which cause the drones to perform some operation on the client side. The drones can also send messages back to the pilot, which cause the pilot to update its internal state and possibly send messages out to its drones.

### 3.7.6 Pilot to Drone Communication

For example, consider an avatar. The pilot would be responsible for providing some user interface that allows the user to navigate through the environment. This might be an external device such as a spaceball or joystick, or it might be as simple as a set of on-screen buttons labelled “forward” and

“backward”. When the user navigates their avatar, the pilot receives the input and converts it into a series of messages for its drones. Those messages get sent out via the networking module, and are received on each observer client. The client takes those messages and simply hands them to the corresponding drones, which take the appropriate actions. This can include asking the graphics engine on the client side to reposition the visual representation of the avatar, or change its pose.
Information can flow in the other direction as well. Consider a virtual water cooler. The user clicks on the water cooler in the 3D window, and that interaction is sent to the water cooler drone running on the user’s client. That drone then calls the user interface to present a selection of choices to the user (such as “pour water” or “drain tank”).
The user selects one of those choices, and the drone sends a message back to its pilot reflecting the change. The pilot updates its internal state (in this case, the water level) and sends messages to all its drones so that they can adjust their visual representation to reflect the reduced water level in the cooler.
In a sense, the networking module provides a central nervous system that relays signals from the brain (the pilot) to the body (the drones), and vice-versa.

### 3.7.7 The Observer’s Perspective

In order to get a better understanding of how the system works, let’s go through it from the standpoint of a new observer just entering the world for the first time.
Upon connecting, the user enters a default location (or a location they specified when starting their client). The observer creates a drone for the location entity, and then tells the drone to initialise itself. The location drone retrieves the visual representation of the location, and builds a scene graph for it. The drone also updates the visual representation of the location, based on state information retrieved from the pilot.
This would include such things as adjusting lighting conditions, setting the open/closed status of doors and windows, and so on. The location drone also provides the observer with a list of passages to other locations, and perhaps a textual description of the location to display on-screen.
The observer also asks the location drone for a list of the entities contained in the location, and then sets up a drone for each of those entities. They initialise themselves, retrieve their visual descriptions, and add them to the scene graph. The entities that correspond to avatars also add themselves to the list of avatars maintained in the user interface module, in order to give the user a list of people in the same location as they are.
If the user enters one of the passages out of the location, the observer client first checks to see if it has access to the location associated with that passage. If so, the observer deactivates the drones for the current location, leaves the location, enters the new location, and goes through the same sequence it did when it entered the current location.

## 4. Synopsis

In this work, we described the generic needs of a groupware educational system, baring in mind the capabilities that the enabling technologies permit. The basic functions determine a set of “could be” attributes but by no means they substitute the educational requirements that the pedagogical experts should define. This work should be looked at as a rapid prototype for the didactical experts to take into account with respect to technological capabilities.
As next steps, the development team will expand the basic prototype, and will include all basic operations. The basic operations that must be supported by the networking module are fairly simple, and should be easy to implement over top of IRC, JSDA or a MUD client. We need to be able to connect to the system, and disconnect from it. We need to be able to enter and leave locations, and

the pilots and the drones need to be able to communicate with each other. Everything else is handled by the pilots and the drones, including text chat, whispering, navigation, and so on.
There also needs to be a registration/authentication mechanism for the pilots to use, in order to allow them to make changes to the world (and even just to manifest themselves visually in it).
Currently we are implementing a preparatory prototype of the system presented. Although not fully functional this prototype has some basic functionality built into it. This prototype will be used as a basis for the end users (students and educators) to experience and evaluate the underlying technology as well as for the consortium to gather and consolidate user requirements for the system, especially with regards to the didactics and educational methodology that will be followed.

## References


1. D. A. Agarwal, S. R. Sachs, and W. E. Johnston.
   The Reality of Collaboratories, *Computer Physics Communications*, 110(1-3):134-141, May 1998.
2. D. Kilman and D. W. Forslund, An International Collaboratory Based on Virtual Patient Records, *Communications of the ACM*, 40(8):110-117, Aug 1997.
3. C. Watters, M. Conley, and C. Alexander, The Digital Agora: Using Technology for Learning in the Social Sciences, *Communications of the ACM*, 41(1):50-57, Jan 1998.
4. Comm. of the ACM, Special Issue on Digital Library, *Communications of the ACM*, 41(4), April 1998.
5. H. Lieberman, N. van Dyke, A. Vivacqua, Let's browse: a collaborating browsing agent, *Knowledge Based Systems 12 (1999) 427-431, Elsevier Press*
6. Grief, Irene, Computer-supported Cooperative Work, Morgan Kaufmann, San Mateo, CA, 1988.
7. "Agent Technology: Foundations, Applications and Markets" by Nicholas R. Jennings (Editor), Michael J. Wooldridge (Editor), Nick. R. Jennings (Editor), ISBN 3540635912
8. J. Bradshaw (Ed.), Software Agents MIT Press, Cambridge, MA, 1997.
9. M. Huhns (Ed.), Readings in Software Agents Morgan Kauffman, SanMateo, CA, 1997.
10. Maes P, Agents that reduce work and information overload, Communications of the ACM, 37(7), 31-40, 1994
11. Tom Stein, Collaborative Training Gains, Information Week Online July 12 1999, http://www.informationweek.com/743/train.htm
12. "Networked Virtual Environments – Design and Implementation", by Singhal, Sandeep and Zyda, Michael, ACM Press Books, SIGGRAPH Series, 23 July 1999, ISBN 0-201-32557-8.
13. VR News, "Shared Worlds Overview", by Bernie Roehl, April 2000.
14. Personal Communication with Roni Aviram & Shahaf Gal, May 2000.
15. C. Davarakis and H. Mayer "Virtual European School - An Infrastructure to Deliver High-quality Contents to Second-level Schools", Online Educa Berlin, December 2-4, 1998.