

# Integrating Pervasive Information Acquisition to Enhance Workspace Awareness

Alois Ferscha  
 Department of Computer Science  
 University of Vienna  
 ferscha@ani.univie.ac.at

## Abstract

*Workspace awareness as the "... up-to-the-moment understanding of another person's interaction with a shared workspace" [GuGr 99] involves knowledge on who is working in the workspace, where individuals are working, what they are doing or going to do, how and when they are executing their work, and what their motivation is for doing it (why). Traditional awareness systems use dynamic user behavior data as collected while monitoring events from I/O devices (keyboard, mouse, touchscreen) at the interface to user. To preserve and maintain a more intuitive fidelity of awareness, we extend our workspace awareness system TEAMSPACE to collect and exploit awareness information from the users physical activities in his workspace, like hand gesture and body movement, using position and orientation tracking technologies. Thus user activities aside the interaction with desktop computing facilities are seamlessly integrated into a shared virtual workspace opening a whole new dimension of awareness abilities.*

**Keywords:** Virtual Spaces, Distributed Cooperative Environments, Multiuser Interaction, VRML, Java, WAP, Tracking.

## 1 Introduction

Providing an intuitive fidelity of staying aware of each other in virtual cooperation environments new means have to be found that support social interaction to a much higher extent than observed today. Most of the user interfaces for multi-user cooperation environments (particularly collaborative software tools) appear inefficient and clumsy as compared to face-to-face cooperations. User helplessness, frustration and denial are the consequences of the awkwardness of many contemporary real-time distributed groupware systems. Supporting pervasive awareness of others, therefore, promises the improving of the usability of multi-user remote cooperation systems by maintaining the fluidity and naturalness of team cooperation.

Looking at existing research on real-time distributed groupware one finds that awareness issues have been considered in isolated and localised situations, and many application or domain specific solutions (like e.g. in WYSIWIS groupware systems [FPP 95][RoGr 96]) are at hand. More generic or general design principles for awareness support systems however have not been proposed yet [GuGr 99][Prin 99]. More principled

information on the awareness oriented design of teamware is demanded, allowing for a systematic consideration of concepts for situated applications and purposes. The focus is on the support of team-work rather than task-work [SPBS 95], i.e. on the issue of how well a system supports activities involving communication, coordination and collaboration among team members, rather than the execution of domain tasks (e.g. the management of ToDo lists) [SPF 99]. In supporting team members to stay aware of each other in a team workspace, the respective teamware systems usability improves, and by that team performance and efficiency [Prin 99].

In previous work we have developed **TEAMSPACE**, i.e. software to configure and operate virtual spaces based on the metaphor of "shared network places" [FeJo 99] as an approach for multiuser, multimedia, distributed cooperative work environments to support the work activities and interactions of goal oriented business teams. Much like real-life physical team rooms, which provide a permanent shared space used by a work group, **TEAMSPACE**, provides for virtual spaces for (possibly mobile) work teams, i.e. a places to store, retrieve and present multimedia content (documents), to design and construct cooperatively, to brainstorm, to negotiate and make decisions, to run shared applications, to browse enterprise data, or to acquire, formalise and exchange enterprise knowledge. **TEAMSPACE** transcends distance, time zones, and organisational boundaries, and integrates pervasive computing approaches to collect awareness information. **TEAMSPACE** is based essentially on two open technologies, the virtual reality modelling language (VRML) together with Java, and the Wireless Application Protocol WAP. A Java-enabled Web-browser together with a VRML plug-in is the only requirement for the execution of **TEAMSPACE** on current consumer hardware. Mobile users equipped with WAP enabled handsets are seamlessly connected to **TEAMSPACE**.

This paper is organised as follows: Section 2 motivates a pervasive awareness systems by outlining some misconceptions of multi-user collaboration software. Section 3 develops a detailed architecture of our open standart Internet collaboratory and explains the major design choices. We demonstrate key features of **TEAMROOM** in a usage scenario in Section 4.

## 2 Development of a Pervasive Awareness Information Framework

What appears to be critically important for the design of virtual workspaces is the conceptual structuring of perceptualisation in team work. Related to the human factors literature [Ends 95], an attempt for such a structuring will focus on

- (i) the “perception of relevant elements of the environment”, i.e. team members must be able to gather perceptual information from the workspace and selectively attend to those artefacts and team activities most relevant for the execution of the current task,
- (ii) the “comprehension of those elements”, i.e. team members must be able to integrate the perceived perceptual information with existing personal and team knowledge and make sense of this information for the current situation, and
- (iii) “prediction of the states of those elements in the near future”, i.e. team members must be able to anticipate changes in the work space and plan accordingly.

The TEAMSPACE awareness information framework identifies the characteristics of team knowledge acquired from observing the dynamics of the workspace (knowledge about the workspace, about the team members and about the work agenda or tasks), and the proactive use of this knowledge to progress team work. The interaction among team members and the workplace itself is explained as a relationship between a team members knowledge and its awareness information gathering activity: observing the workspace will modify the preliminary knowledge about it, thus direct new exploration activities, which in turn will improve the knowledge. Workspaces awareness in this context hence requires the consideration of how participants should interact with each other in a shared space, how co-manipulated objects should behave, how the collaborators should be represented in the collaborative environment, how to effectively transmit non-verbal cues that real-world collaborators can use effectively, how to transmit video and audio via channels that allow both public addressing as well as private conversations to occur, etc. We summarize the following driving motivations for the development of **TEAMSPACE**:

- The most general and most commonly understood metaphor for team work is the notion of a "meeting", symbolising a "come together" of team members, a common goal, the use of means to reach the common goals, to exchange facts and thoughts, to prepare and make decisions, to develop and progress workplans.

TEAMSPACE aims to consequently follow the "meeting room" metaphor as a methodological approach of integrating memory and awareness with a high level of "**immersiveness**". (Immersiveness, here, relates to the degree to which team members mentally associate themselves with the "coming together" – often described as a mental status of "feeling like being there without actually being there".)

- Meeting support today is mostly understood as the synchronous communication among team members [McHo 94], like e.g. desktop video conferencing systems can support geographically dislocated teams, while asynchronous communication (e-mail, workflow management, etc.) are understood and deployed orthogonally. We aim at a **seamless integration of synchronous and asynchronous interaction** (not necessarily just communication) means as the convergence of augmenting asynchronous communication with synchronous components (e.g. structuring e-mail discussions with response deadlines) and vice versa (e.g. augmenting workflow with real time interaction mechanisms).
- Proprietary CSCW software is limited in support functions, bound in team size, and often isolated from the business or team context. Cooperation tools supporting meetings must be **customisable** to different kinds, types and sizes of teams (persistent teams, ad-hoc teams), must be **adaptive** to changing team business processes, work agenda, roles of team members, team size and even user preferences, and **integrate** different support services (calendar, brainstorming, planning, negotiating, decision making, etc.) and legacy software (Outlook, TeamBoard, NetMeeting, etc.).
- Traditional meetings tend to suffer from bad preparation and insufficient follow-ups. In the TeamWorkplace view, meeting support needs to span the **whole meeting lifecycle** starting from the announcement of agenda, the posting of "a-head" material, the invitation and appointment making, the execution of the meeting itself with all of its integrated subprocesses, and embedding in team memory, preparation, and ending at the automated generation of meeting notes (from video or voice conferences), the indexing, annotation and archiving of the notes. Meeting process cycles, i.e. the linking of one meeting to the other are to be supported, and asynchronous meeting spaces are demanded as a means of **embedding team memory** that can be accessed from any place and any time by any team member – irrespective of a meeting being in progress or suspended or not limited in time.

The main challenge for the development of **TEAMSPACE** was the provision of team memory and team awareness enabled meeting support systems, in the software instance of media spaces providing virtual teams with **real life meeting fidelity**, a **large scale availability** and **universal access**. Particularly access and availability is forcing a radically new approach opposing e.g. multi-user collaborative virtual environments as coming from virtual reality research [WaBa 97] [RCRW 98] (like e.g. Massive [GrBe 95], DIVE [Hags 97], COVEN [COVE] or CAVERN [CAVE]), which by nature of the highly specialised hardware and software components involved prohibit user access on a broad scale. To this end, the employment of standard and open internet technology appears mandatory, hence we committed to the realisation of **IP based Media (Meeting) Spaces** –or in other words– **Internet Collaboratories**. Finally, to support a “real-life” fidelity of coworkers in a shared space, we employ position and orientation tracking technologies to enable a pervasive awareness system.

### 3 Implementation

TEAMSPACE is conceptually developed as shared network places to support distributed cooperative work activities and virtual teams. The TEAMSPACE is an arrangement of one or more virtual rooms (VIRTUALROOM), which can be used for different work tasks by different team members at different times and for varying purpose. Each of these rooms can have different characteristics tuned on the special needs of the work done therein. TEAMSPACE is based essentially on two open technologies on Java [ChCo 97] [CiRo 98] together with the virtual reality modelling language (VRML [HMRR 97]) and WAP. In its software architecture, TEAMSPACE follows a centralized server approach. Due to the security policy of most Web browsers (e.g. Netscape Navigator or Microsoft Internet Explorer), which allows applets to connect only to the server from which the applet itself was downloaded (this basically precludes all other network communication topologies like multicast and peer-to-peer communication), a client/server communication structure had to be adopted. Although these security constraints can be circumvented through the use of insecure browsers, further considerations also favor a client/server communication structure. First, the VRML world and its current state must reside (or be stored) somewhere on the Internet when no users are currently in the world. This is most conveniently achieved through the use of a dedicated server. When the server is running, the current state of the world resides in the memory of the server. When new participants join, the server simply has to send descriptions of all active objects (VRML files) and their current state to the new participant. In the case that the server must be shut

down, the VRML world and its current state can be saved to the server’s storage.

Of course, client/server architectures do not scale well

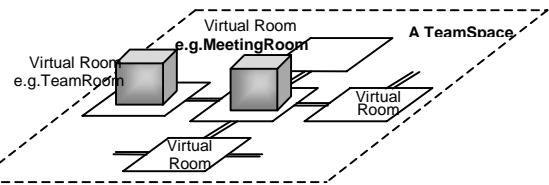


Figure 3: A TEAMSPACE and Virtual Rooms

to a large number of users since the server soon becomes the communication bottleneck due to the overwhelming amount of messages which must be forwarded [Bla] [AWS]. In typical cases, the number of simultaneous users in client/server DVEs has a practical limit in the range of tens of users, with loss in performance increasing as the number of users grows [Bro 98][Prin 99].

#### 3.1 General architecture

The TEAMSPACE is divided into two main parts:

**The TEAMROOM:** A virtual 3D representation of the work environment (In our case an exact representation of the second floor of the Computer Science Department). Where each team member has a workplace which corresponds to the workplace in the real environment. This part is mainly the awareness aspect of the TEAMSPACE. It shows which members are currently online and represented by their avatars. In some cases the activity of the user is shown. This is achieved through the use of tracking system(s) that map the member’s position to certain activities. It is also possible to send notification messages, SMS or files to

**The MEETINGROOM:** A special virtual 3D MEETINGROOM located in the center of the TEAMSPACE, where virtual meetings of Team members can be held. Here documents can be viewed or slide shows are presented to other members in a meeting.

##### 3.1.1 General Concepts

In order to create a stable and consistent multi user environment a client/server architecture is used where the server holds the current state of the TEAMSPACE, receives changes to the TEAMSPACE done by a member and sends these changes to all connected clients. In our case the clients are presented by a VRML world that represents the TEAMSPACE. This VRML world is embedded in an html page together with an embedded applet that controls the objects in the VRML scene via the before mentioned EAI (external authoring interface). When a client using a Web



Figure 4: A TEAMSPACE and Virtual Rooms

browser connects to the server the static VRML world is loaded (all objects that can not be cooperatively manipulated in the VRML World). After that the embedded Java applet makes a socket connection to the server and all active objects are loaded (objects that can be manipulated cooperatively). The client applet then connects to the VRML world and creates the VRML objects in the world. A more accurate description of active objects follows below. In addition to that, the client applet assigns an observer to each eventout of each active object in the scene to monitor the changes done to the objects.

### 3.1.2 Active Objects

We have developed special objects in our TEAMSPACE which are able to transmit the actions through the server to other clients all by themselves. With this architecture new interactive objects can be easily introduced to the existing environment. This section describes the functionality and implementation of these objects.

### 3.1.3 VRML Objects

In the case of a single-user VRML world, the entire description of the world (i.e. the “scene graph”) is typically contained in one VRML file (with the extension “.wrl”). In the case of dynamically changing VRML worlds, however, this type of information management is not suitable since objects are typically added, removed or their attributes changed during the lifetime of the world. Furthermore, since objects can send and/or receive events, the controlling application must be able to identify objects and access them individually. This necessitates some sort of indexing or naming scheme for objects so that events which are received can be expediently routed to the correct object in the VRML scene.

For this reason, the VRML scene is considered to consist of a set of self-contained objects. Furthermore, interaction with objects is only possible through the event interface of

their top-level node. In VRML, of course, any constituent node of a (compound) object (which is in fact a compound node itself) may be addressed individually through the DEF mechanism. Owing to the fact, however, that the EAI cannot access node by their DEF names, this type of node addressing is not possible. At first glance this would seem to be an unfortunate restriction but in fact this type of addressing is not necessary at all.

In order to deal with objects of arbitrary structure and functionality, it is necessary to standardize the way in which other objects and users can interact with objects. This can be achieved by moving all events (as well as fields and exposedFields) which should be accessible from the outside world into the top-level event interface of the object. For implementing this type of object encapsulation, VRML provides a convenient means: the prototype definition (PROTO). The prototype header consists of the word “PROTO” followed by the name of the prototype and the event interface which is enclosed in square brackets.

VRML distinguishes four types of events which may appear in the event interface of nodes and prototypes:

1. The `field` keyword defines events which can only be received upon initialisation (instantiation) of the object and cannot be subsequently changed during the life of the object.
2. The keyword `eventIn` defines events which the object is capable of receiving (but not sending) and
3. `eventOut` defines events which the object can emit (but not receive).
4. If the value of field can be changed at run-time, VRML provides the event type `exposedField` for defining fields which are capable of both sending and receiving events.

Prototypes are stored by the server in the prototype database. Clients wishing to import new objects into the shared world can send new prototype definitions to the server at any time. The server, after checking for naming collisions among the objects, then inserts (or replaces) the prototype definition in the database where it can then be referenced and sent to other connected clients.

Prototype definitions, however, do not define actual objects in the VRML world but provide only a “blueprint” for objects. In order to actually create an object and place it in the world, a prototype must be instantiated with actual values. VRML provides two means for instantiating prototypes. The first method, which is practical only in single user worlds, allows for the prototype definition to

be included directly in the VRML file containing the entire scene. Instantiations of the object may be created simply by using the prototype's name as a node name (the prototype actually defines a new type of node). The second method, which is also applicable for dynamic multiuser worlds, permits the use of object definitions which are not included in the main VRML scene file but in external files (which the VRML browser then may reference as a typical URL). These are included in the main scene as so-called EXTERNPROTOS.

The TEAMSPACE server, besides maintaining a prototype database, also maintains a database of currently existing object instances. The file format for these objects contains an EXTERNPROTO definition, followed by a node instantiation. Furthermore, the information specifying which events must be propagated in the multiuser world are appended in the form of routing information at the end of the file.

When processing an instantiation file, the server creates an instance of the object in its database of existing objects and assigns a unique numerical object identifier to the new object (objectID). The BROADCAST keyword gives information for broadcasting outgoing events in the multiuser world and is modelled after the standard ROUTE syntax in VRML. The reserved word "this" is used to designate current object. Numerical event identifiers (eventID) are assigned to each event sent by the object beginning with 1 (in the order of the BROADCAST lines). Thus, each (outgoing) event can be uniquely identified using the objectID and eventID. The server then forwards the instantiation file to all connected clients.

The clients, in their turn, pass the EXTERNPROTO definition and the following object instantiation "as is" to the VRML browser via the EAI. This immediately creates an instance of the object in the VRML world. The information contained in the BROADCAST lines is used to set up the callbacks which detect event occurrence in the VRML world and create references to the object and its events which are used for passing incoming events to the object.

### 3.1.4 Creation of Active Objects Serverside

To keep a consistent representation of the TEAMSPACE we decided to create Java representations of the VRML Objects that are maintained by the Server. When the server starts, it reads all VRML object in the objects subfolder. There it finds EXTERNPROTO like this cupboard in the MEETINGROOM:

```
EXTERNPROTO Kasten [
  field SFString owner
  field SFInt32 poscount
  field SFVec3f pos1
  field SFVec3f pos2
  field SFVec3f pos3
  field SFVec3f translation
  field SFRotation rotation
  eventOut SFBool doorclicked
  eventOut SFBool corpusclicked
  ExposedField SFBool enabled
  eventIn SFBool dummy
] "protos/Kasten_neu2.wrl"

DEF Kasten1 Kasten {
  owner "scheiner"
  poscount 3
  pos1 -1.0 6.75 1.0
  pos2 -1.0 4.95 1.0
  pos3 -1.0 3.25 1.0
  translation -15 0 -14
  rotation 0 1 0 0.6283
  enabled TRUE
}

BROADCAST THIS.doorclicked TO THIS.dummy
BROADCAST THIS.corpusclicked TO THIS.dummy
```

Interfacedescription (Eventinterface)

Address of the Prototype

Instance of Prototype

Broadcasting

To check the objects when creating Java representations we developed the VRMLTokenizer. This class performs several syntax checks parses the EXTERNPROTO declaration of the object and the Instantiation of the object and creates the Java representation. In addition to that, during the parsing of the objects some special flags are set to make identifying the objects easier. These flags are:

- isContainer: this Object is a cupboard
- isMedia: this Object is either a document, a slide show or a movie in a cabinet
- isPlayer: this Object is the MediaPlayer in the MEETINGROOM
- isAvatar: this Object is an Avatar in the MEETINGROOM
- isIndicator: this is the Indicator Object
- isEntryway: clicking this object invokes the login procedure to the MEETINGROOM
- isPinwall: this object is one of the Pinwalls in the TEAMROOM
- isAblage: this object is one Filefolder in the TEAMROOM
- isProperty: when this object is clicked the properties of one participant in the Meeting can be changed
- isReload: these Objects have to be reloaded when a new person enters the MEETINGROOM
- isAwareAvatar: this object is an Avatar in the TEAMROOMS

In addition to the active objects created during the Serverstart there are some objects created dynamically during runtime. The next list shows all those objects and their time of creation, if they have to be reloaded and if they are loaded from the objects directory or created dynamically by the server:

Name (description)	Time of creation	Origin	Reload
Floor (of MEETINGROOM)	Serverstart	Objects	No
Computer (in MEETINGROOM)	Serverstart	Objects	Yes
Indicator (in MEETINGROOM)	Serverstart	Objects	Yes
Cabinet 1-4 (in MEETINGROOM)	Serverstart	Objects	Yes
MySeat (of one participant in MEETINGROOM)	During login in MS	Objects	Yes
Officechairs (in MEETINGROOM)	Serverstart	Objects	No
Phone (in MEETINGROOM)	Serverstart	Objects	Yes
Plant (in MEETINGROOM)	Serverstart	Objects	No
Tables (in MEETINGROOM)	Serverstart	Objects	No
Television (in MEETINGROOM)	Serverstart	Objects	No
Pinwall (in TEAMROOM)	Serverstart	Server	No
Filefolder (in TEAMROOM)	Serverstart	Server	No
Avatar (in MEETINGROOM)	During login in MS	Server	No
AwareAvatar (in TEAMROOM)	During login in TR	Server	No
Media (object in one Cabinet)	User created	Server	No

After the EXTERNPROTO declaration and the instantiation of a VRML object are parsed and converted, the Routes of the Object are parsed. These Routes are marked with the word BROADCAST and show which eventouts of the VRML object should be monitored. The Route is then converted to a Forwarder object that represents the route. Each VrmlObject has its own vector of Forwarder objects corresponding to the real VRML object. One Forwarder object is just a forwarding mechanism that receives state changes from the client and relays them to all other clients. When the VRML object has passed all syntax checks it is assigned a unique ID and it is saved in a hashtable for easy retrieval.

### 3.1.5 Creation of Active Objects Clientside

The client itself only creates active objects if he gets them from the server. The creation process is quite similar to the creation at the server with two differences:

- When the Client receives a string from the server describing the active object, it creates the Java VrmlObject and a real VRML node in the VRML scene. The server also sends the ID of the object to

ensure that the object has the same ID as the same object on the server.

- Instead of adding a Forwarder object to the VRMLObject a Route object is created for each broadcast statement found in the VRML String. This Route object is more complex than the Forwarder object. It observes the eventOuts of the VRML node and performs different actions according to the specified behaviour of the VRML node.

The creation of the VRML objects is depicted in Figure 5

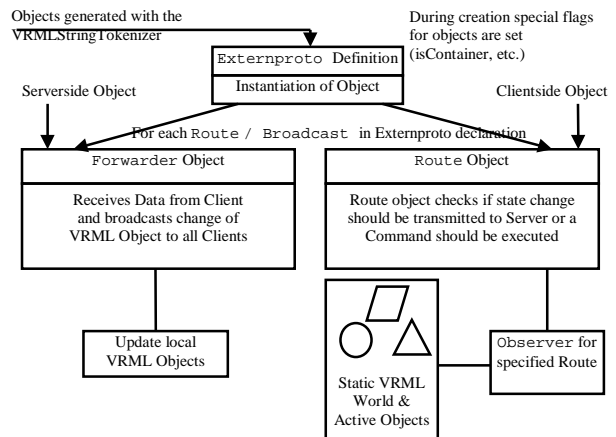


Figure 5: Creation of Active Objects

### 3.1.6 Forwarding / Routing in TeamSpace

The proposed architecture allows for an easy integration of new active VRML objects. add a new active object to the environment the new object must be introduced in the specified EXTERNPROTO declaration and the Route object for the monitored events must be changed. The process of relaying state information from one client to all others is shown in Figure 6.

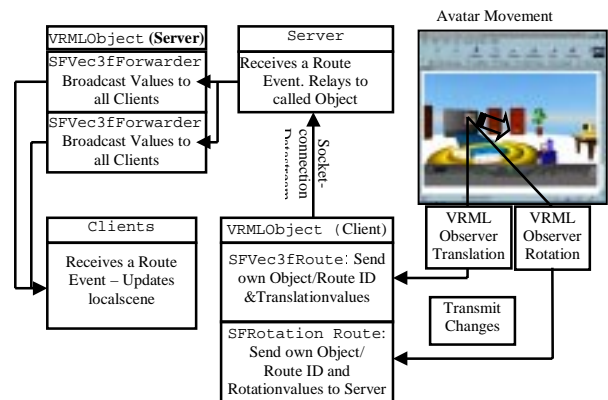


Figure 6a: Forwarding / Routing in the TEAMSPACE

When a member in the MEETINGROOM changes his position the Avatar representing him moves also in the VRML scene. To relay this movement to all other clients, two observers are spawned, when the Avatar is created: One observer that tracks the rotation and one that tracks the translation of the Avatar. These changes are transmitted to the route object of the corresponding VRML object. The route object identifies what it to do with the caught event:

- Send the state change to the server
- Perform a special command (like putting a note on someone's pinwall)

In our case the state change should just be sent to the server. This is done via the socket connection the client has established when it logged in. Using a Java `DataStream` it sends the object ID, route ID and state change to the server. The server is always listening for input from its clients and differentiates the requests by the ID sent from the client. When a client sends a request starting with a positive Integer, it assumes that a state change is coming in and hands the rest of the communication to the VRML object and the Forwarder object, which are identified by the IDs sent from the client. If the first ID is negative a special command is requested (like send the whole VRML scene to the client). In our example the server receives the positive IDs of two state changes and the forwarder object sends these state changes to all other clients.

As the amount of data transmitted can be very large (especially in this case where every state change would have been transmitted), we use intelligent routes to avoid heavy net traffic (Figure 7).

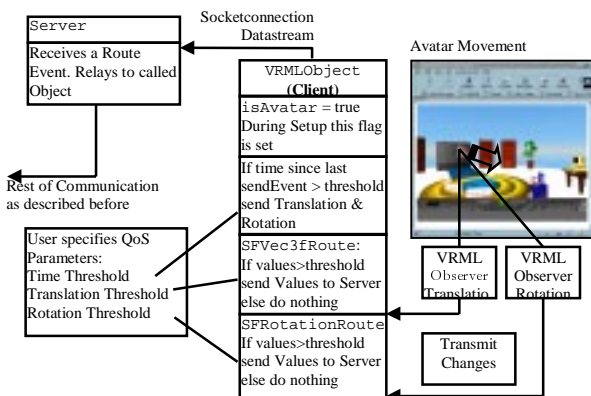


Figure 6b: Performance Optimisation for Forwarding / Routing

When the Avatar of the member is moved, the assigned observers send the state change to the Route object of the corresponding `VrmlObject`. The Route object knows

that it is getting data from the Avatar and sends the data to the server only if the translation or the rotation is greater than a specified threshold. We decided to leave this task to the client to take some computational load off the server and some traffic off the network. To show one of the many uses of the intelligent routing we implemented intelligent routes for the Avatar movement.

### 3.1.7 Server Architecture

The Server is a multithreaded Java applications that has three main tasks (Figure 8):

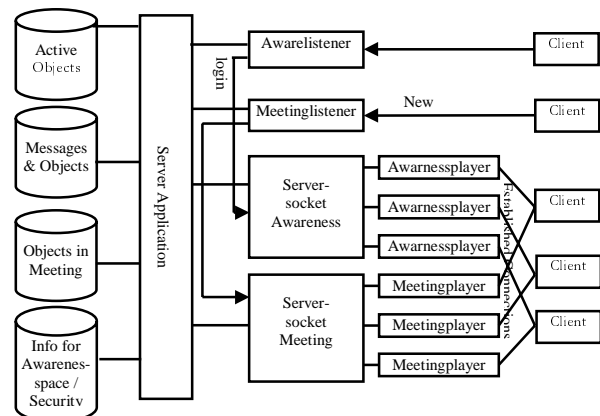


Figure 7: Server Architecture

1. Administrate relevant information in TEAMSPACE:
  - hold and update all active objects in the TEAMSPACE
  - manage the messages on the pinwalls and the objects in the filefolders in the TEAMROOMS
  - keep track of all objects in a meeting
  - administrate security information
  - wait for incoming new connections
  - The two Listenerthreads are listening for new connections and assign new Meetingplayer and Awarenessplayer to connections that log in correctly. We divided the communication for the TeamRooms and the MeetingRoom to take load from the different ports and to have a clear separation of these two communication streams
3. Communicate with all clients that have already established connections
  - The player objects perform all tasks that are necessary to operate the TeamSpace. They relay relevant data to all clients and perform special commands (sending the whole world to a client or saving a new message for a member).

### 3.1.8 Client Architecture

The client consists of the embedded VRML scene and the embedded client applet on a HTML page. The client applet

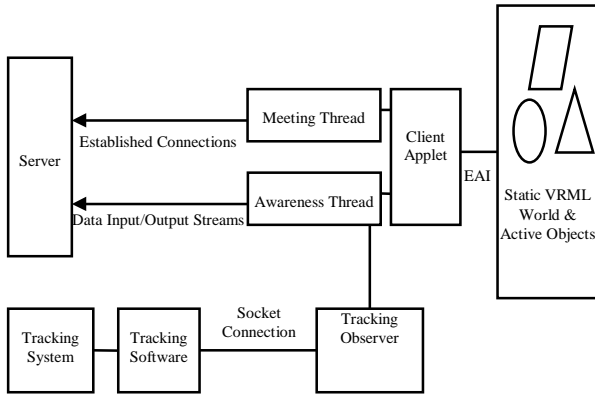


Figure 8: Client Architecture

connects to the server via the two communication threads (Meeting Thread and Awareness Thread) and communicates with the server via a socket connection using a standard Java Data Input / Output Stream. A connected user can publicise his position information in physical space by using a tracking system (in our case a 6-DOF magnetic position and orientation tracking system is used to track the users hands and hip position relative to the “spheres of influence” of physical office equipment, like e.g. keyboard, telephone, book shelf etc. – see Figure 9), or by the viewer position of his VRML browser. TEAMSPACE itself detects if the user is wearing the tracking system sensors and starts the tracking observer, which itself opens a socket connection to the server. Translation and rotation information to control the movement of the avatar representing the user in the TEAMROOM is thus transparently collected from either the VRML browser (viewpoint node) or the tracking software.

4

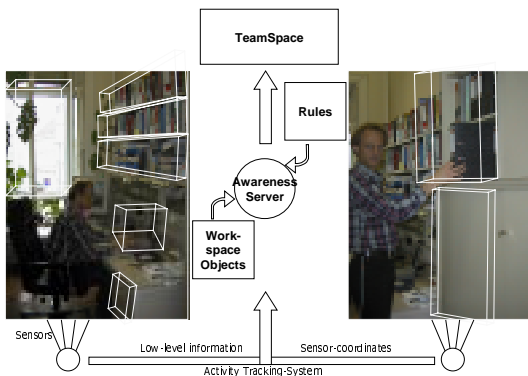


Figure 9: Magnetic Position/Orientation Tracking

### TeamSpace Showcase

As a demonstration showcase, we have constructed a virtual meeting space for workgroups on top of the TEAMSPACE software architectures, following an office metaphor for environment visualization (Figure 10).



Figure 10: Birds view to MeetingRoom

The TEAMROOM is mainly devoted to provide workspace awareness to users: department members currently in their office can “see” each other and inspect their “availability for interaction” status – even through “walls”.

As shown in the scene above (Figure 11), two department



Figure 11: Team members in the TEAMROOM

members are currently logged in (present in their physical workplace), and the availability status (“ONLINE”) indicates that they are busy working (at their desktop computers). One can also see the users pinwalls and filefolders located in the workspace of each member – in the scene represented by multifunctional image avatars.

If the avatar image is clicked, the UserCard (or homepage) of that user is displayed (Figure 11). Above the avatar image the name and the current online status are displayed. The buttons in the lower area of the avatar image allow members (from left to right) to view the business card of the user, to send messages to the user (or to view the own messages if the own avatar was clicked), send SMS to the user, send objects to the user (or to view the own messages if the own avatar was clicked) and to send emails to the user.

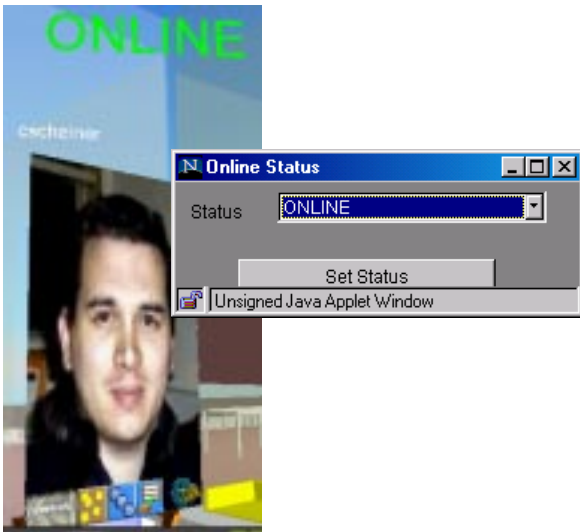


Figure 11: multifunction avatar

The “Online Status” menu is displayed whenever the user wants to change his current status. In the example below user “cscheiner” changes his online status to AWAY, which is then displayed in the TEAMROOM.



Figure 12: pinwall messages

The status information also indicates whether the user has left his workplace and cannot be contacted directly. User

“hlavac” in the example decides to leave a message on user cscheiner’s pinwall to set up a meeting in the virtual TEAMROOM with him. He clicks either the users pinwall or button on the avatar and the following dialog box is displayed (Figure 12):

Leaving a message on somebodys pinwall is displayed in the scene in a real life metaphor (Figure 13), and can be handled via direct manipulation (click to open, drag for moving to filefolder, etc.)



Figure 13: Visualisation of Notes in TEAMROOM

To support mobile team members without permanent connection to TEAMSPACE, a WAP (wireless application protocol) interface has been implemented to allow roaming users to register and login, to receive, view, answer and delete messages, to maintain contact list (create, add user, delete, rename, remove user), to inspect UserCards and to access user availability information – all via a WAP enabled handset (typically mobile phone).

## 5 Conclusions

Multiuser distributed cooperation environments are one of the fastest growing industrial fields of application of the Internet [WAS 97], particularly its most popular service, the WWW. It is becoming an indispensable vehicle for the interconnection of humans in the interactive execution of their cooperative work agenda.

This paper has exemplified the organisation of mobile virtual teams interacting “on the move” and across geographical boundaries, and has developed a workspace awareness solution for a distributed cooperation environment, exclusively based on standard Internet technologies. The commitment to open standards was the consequence after systematically pointing out the

misconception of proprietary, platform and application dependent cooperation software. Hence, an integrated software architecture for a distributed cooperative work environment has been developed – the TEAMSPACE – and key features of such an environment have been implemented as a proof of concept. The key design choices made for this project were the following:

- TEAMSPACE is exclusively based on standard Internet technologies, like HTTP, VRML, JAVA, TCP/IP and WAP. With this approach we are able to support any kind of desktop platform and thus provide transparent access from whichever user terminal, and from wherever it connects to the Internet.
- To implement workspace awareness, the TEAMSPACE software architecture is based on a central server exploiting the capabilities of the platform independence of Java virtual machines, and Java enabled browsers to implement Internet clients. Mobile client terminals like PDAs, smartphones or GSM and WAP enabled handsets are supported using Push-services.
- Fully distributed systems pose the problem of providing dedicated entry points for new users, i.e. to allow new participants to connect to a certain server port on the machine executing the server application. TEAMSPACE, with a single, centralized entry point for new participants avoids the difficulties of joining posed by distributed protocols altogether.

The modular concept and structured interfaces of the TEAMSPACE software architecture allows for easy extensibility and adaptation to special needs. The integration of new TEAMSPACE components (“rooms”) is eased, thus making TEAMSPACE future proof with respect to new functionality and changing technology. Presently, TEAMSPACE integrates the modules MEETINGROOM and TEAMROOM.

The two main components of the TEAMSPACE environment are the MEETINGROOM, a virtual place to gather and perform virtual conferences, and the TEAMSPACE, an awareness system, which provides the user with up-to-date information about the status of co-workers (team members) or the work process itself. The main characteristics and most important features of the TEAMSPACE are:

- The use of VRML for the design of the TEAMSPACE enables an intuitive 3D environment. The natural look and feel of the user interface makes it easier for the user to interact with the environment and co-workers as compared to flat, window-based interfaces.

- The simulation of a real world environment like e.g. dynamic position visualisation allows for a wide variety of real-life meeting situations.

In our work we have created software components, which in a modular way provide core functionality to the user. Referred to as “active objects”, these components have individual visual appearance (like cupboard, pinwall...) in the MEETINGROOM:

- The “container” object acts as a persistent repository for other objects, and displays its content to the user. A container like the cupboard as an example provides space to deposit other objects, which in turn can be the subject of direct user manipulation.
- Objects deposited in “container” objects are referred to as “media” and have instances as 3D representations for video, document or a slide show objects. Not only can these media be stored in a “container” object, but also be “brought” into the MeetingRoom by a user by just providing the respective unified resource locator (URL).
- A dedicated “player” object provides the MeetingRoom with the functionality to retrieve or view the documents of the “media” type according to a real-life metaphor user interface.

The second main part of our environment is the TEAMROOM (presented in this paper) which acts as a workspace awareness system. The main focus in the TEAMROOM is the collection and delivery of awareness information from and to the virtual environment, so as to make team members aware of their physical presence and thus implicitly announce their status of availability for interaction

- In TEAMROOM, awareness is supported first through the 3D simulation and visualisation of a real world metaphoric space, e.g. a department, office etc., and secondly through the use of user representations, so called avatars, at any time displaying the physical presence, position and orientation of each team member currently participating in the system.
- Position and orientation information to control the movement of avatar representations of users is seamlessly collected via the VRML browser viewpoint settings, or the tracked position of a user acting in his real workspace .
- Specialized active objects allow for leaving and receiving messages, post or read notes, send or receive notifications to mobile phones (SMS) or to acquire/change user status or authentication information are provided in TEAMROOM.

## 6 References

- [AWS] Active Worlds Servers. [www.activeworlds.com](http://www.activeworlds.com)
- [Bla] Community Server. Blaxxun Interactive. [www.blaxxun.com](http://www.blaxxun.com)
- [Bro 98] W. Broll, DWTP – An Internet Protocol For Shared Virtual Environments, Proceedings of the International Symposium on the Virtual Reality Modeling Language 1998, ACM, pages 49-56, 1998.
- [CAVE] CAVERN: <http://www.evl.uic.edu/spiff/covr/>
- [ChCo 97] M. Chen and J. Cowie, Java's Role in Distributed Collaboration, Concurrency - Practice and Experience 9, 6: 509-520, 1997.
- [CiRo 98] P. Ciancarini and D. Rossi, Coordinating Java Agents Over the WWW, World Wide Web, 1(2):1998.
- [COVE] COVEN: <http://chinon.thomson-csf.fr/projects/coven/>
- [FeJo 99] A. Ferscha, J. Johnson: Distributed Interaction in Virtual Spaces. Proc. of the 3<sup>rd</sup> International WS on Distributed Interactive Simulation and Real Time Applications. IEEE CS-Press, 1999.
- [FPP 95] Fuchs, L., U. Pankoke-Babatz, W. Prinz, "Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System", in Proc. of the European Conference on Computer Supported Cooperative Work (ECSCW '95), Stockholm, Kluwer, 1995, pp. 247-262.
- [GrBe 95] C.M. Greenhalgh, S.D. Benford: MASSIVE: A Virtual Reality System for Tele-conferencing. ACM Transactions on Computer Human Interfaces (TOCHI), 2 (3), ISSN 1073-0516, ACM Press, September 1995, pp. 239-261.
- [GuGr 99] C. Gutwin, S. Greensberg: A Framework of Awareness for Small Groups in Shared-Workspace Groupware. Technical report 99-1, Department of Computer Science, University of Saskatchewan, Canada: <http://www.cpsc.ucalgary.ca/grouplab/papers/>
- [Hags 97] O. Hagsand: Interactive Multiuser Ves in the DIVE System, IEEE Multimedia, Vol. 3, No. 1 , 1997, pp. 30-39.
- [HMRR 97] Y. Honda, Mitra, B. Rockwell, B. Roehl, Living Worlds – Making VRML 2.0 Applications Interpersonal and Interoperable, Draft 2.0, 14. April 1997, [http://www.vrml.org/WorkingGroups/living-worlds/draft\\_2](http://www.vrml.org/WorkingGroups/living-worlds/draft_2)
- [Prin 99] Prinz, W. "NESSIE: An Awareness Environment for Cooperative Settings", in Proceedings of The Sixth European Conference on Computer Supported Cooperative Work - ECSCW'99, S. Bødker, M. Kyng, and K. Schmidt (eds.), Kluwer Academic Publishers, 1999, pp 391-410.
- [RCRW 98] G. Reitmayr, S. Carroll, A. Reitemeyer and M.G. Wagner, DeepMatrix - An Open Technology Based Virtual Environment System, ASU-CSE-TR-103098, The Visual Computer Journal, [vienna.eas.asu.edu/wagner/academic/papers/deepmatrix.html](http://vienna.eas.asu.edu/wagner/academic/papers/deepmatrix.html)
- [RoGr 96] M. Roseman, S. Greenberg: Building Real-Time Groupware witj GroupKit, a Groupware Toolkit. In: Transactions on Computer-Human-Interaction, 3(1), 1996, pp.66-106.
- [SPBS 95] E. Salas, C. Prince, D. Baker, L. Shrestha: Situation Awareness in Team Performance: Implications for Measurement and Training. In: Human Factors, 37 (1), 1995, pp. 123-136.
- [SPF 99] Sohlenkamp, M., W. Prinz and L. Fuchs. "PoliAwac - Design and Evaluation of an Awareness Enhanced Groupware Client", AI and Society - Special Issue on CSCW 1999.
- [WaBa 97] R.C. Waters, J.W. Barrus: The Rise of Shared Virtual Environments. In: IEEE Spectrum, Vol. 34, No. 3, 1997, pp. 20-25.
- [WAS 97] R.C. Waters, D.B. Anderson, D.L. Schwenke, Design of the Interactive Sharing Protocol, Postproceedings WETICE 97, IEEE Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, MIT, June 1997, IEEE CS Press, Los Alamitos CA, [www.meitca.com/opencom/istp.html](http://www.meitca.com/opencom/istp.html)