ABSTRACT
In the context of CSCW (Computer-Supported Cooperative Work), we propose to ease the interaction between users through the use of user-aware agents. The purpose of these agent is to be aware of the user's state (e.g., is the user typing on the keyboard, meeting with other people, on the phone, etc.). We will first describe an application we developed on top of a Mediaspace, EasyMeeting, based on user-aware agents. Second, we will present the implementation (multi-agent architecture, language). Finally, we will discuss various aspects of the agents. We believe user-aware agents are a step toward better communication man-machine. Instead of the usual approach in which users consciously interact with the machine, we make the computer aware of the user's actions whereas the user is not aware of the computer listening to his actions. We chose a passive input approach since we did not want to add a burden to the user (e.g., carry badges, fill out forms). This way, we hope the user will benefit from the system without having extra work to do, which avoids a failure of groupware systems pointed out before [9].


INTRODUCTION
The keyboard and mouse are today's computer's main source of input. Considering the human capabilities and the technology available, this is quite limiting of what could be done to improve human-computer communication. In this paper, we propose to add an input source to the computer, the user. We believe this is achieved by providing an "eye" and an "ear" that enable the computer to be aware of what is happening in a particular place. As opposed to the traditional use of keyboard and mouse inputs, this input is of a passive form since the computer is aware of the user's actions. This approach removes the burden of having to use special input devices. We chose this approach since we did not want the user to add new hardware to the computer. This way, we hope the user will benefit from the system without having to do extra work, which avoids a failure of groupware systems pointed out before [9].

We call the agents that listen to the user's physical activity user-aware agents. For example, we can imagine a simple screen-saver utility that would do its job automatically whenever the user is not working in front of the computer. However, this work focuses on the CSCW (Computer-Supported Cooperative Work) field. Indeed, knowledge of the user's state can be quite valuable in CSCW to help improve communication between users. We investigated the use of such agents in a Mediaspace, a multimedia system allowing workers to communicate through audio/video. In this paper, we will describe EasyMeeting, a meeting application on a mediaspace, the implementation in particular the architecture, and discussion of uses of agents, and privacy issues.

RELATED WORK
Both the VideoDesk [11] and the DigitalDesk [24] link the real world to the computer world, by enabling video analysis. In the VideoDesk, a camera is mounted on top of the physical desk and analyzes the scene. It can recognize text (using Optical Character Recognition (OCR) techniques) or text written on a paper. The VideoDesk recognizes the user's hand movements in order to manipulate objects on the screen without using a mouse. This way, we believe user-aware agents are a step toward better communication man-machine. Instead of the usual approach in which users consciously interact with the machine, we make the computer aware of the user's actions whereas the user is not aware of the computer listening to his actions.
Multi-agent systems are related to our work except that our focus is more on Computer-human interaction.

On the mediaspace aspect of our work, many systems exist that study the interaction through video. For instance, Portholes uses the mediaspace in a synchronous way to develop group awareness. Snapshots from people's offices are taken at a regular interval and displayed as a background mosaic picture. Thus, co-workers unconsciously develop a stronger feeling of co-presence. Portholes' approach is different from EasyMeeting in the sense that it brings to the user knowledge of who's in the office and who's not, but does not bring this knowledge to the computer. This could have been done, for instance, by some image processing on the mosaic. In a similar fashion, VideoWindow connects two distant rooms through a large video-display, thus improving communication between distant sites.

Finally, the concept of active office and reactive environment are the most closely related to our work. The active office builds applications based on knowledge of the location of people wearing active badges. Our systems differ in the sense that they do not want users to wear any devices and we also have a type of agent that is not only location oriented.

EASYMEETING: USER-AWARENESS IN A MEDIASPACE

Part of every day's office work often consists of informal/formal meetings between co-workers. However, those meetings are often informal and do not happen easily. As an example, consider a situation where A wants to meet B. User A dials B's telephone number and gets a busy tone. A moment later A passes by B's office and finds the office closed. A then leaves a note asking B to get back in touch with him. When B returns, a similar scenario happens where A is out of the office making it difficult for B to reach him. This scenario can happen indefinitely!

Although reality is not so negative, people in offices still spend lots of time running after each other to improve communication. It is often easier to improve communication through face-to-face meetings. For this purpose, we developed EasyMeeting, an application that allows people to meet easily.

In the next section, we will define what is a mediaspace and then describe the EasyMeeting application (office metaphor for the user interface features...).
fits better the needs of CSCW. The same way single-user interfaces based on the desktop metaphor seemed natural extend it, in the case of CSCW, to an office metaphor. Office buildings have three basic spaces where people meet. First, in their office, where formal "serious" meetings occur. Second, in the hallway, where co-workers can glance in each other’s office and hold informal meetings. Finally, in the coffee room, where informal meetings take place whether work or non-work related.

From the user-interface’s point of view, the metaphor directly represents three rooms representing various levels of communication from formal communication (office) to informal communication (hallway and coffee room):

Office room
This means that users want to meet privately from the agent’s point of view, the users are in their office, alone and not on the telephone.

Coffee room
This is an informal type of communication. Users want to meet, even in a public place. For instance, if the system detects user A in a public place and user B at his office, they will still be connected, even though there might be people around.

Hallway room
Provides the ability for users to glance in someone else’s office unless the user explicitly sets his/her access door. This is always possible. The connection lasts only a few seconds.

The user interface has been designed so that meetings are very easy to obtain. Users are represented as icons when they want to meet someone. When the system detects a meeting, users are represented as icons when they want to meet someone. Users can drag and drop icons into the appropriate room (optionally, he/she can set a later meeting time). Each kind of meeting is represented by a different icon. The system guesses what the user is doing and what kind of connections he/she wants. For instance, if a user is meeting people at his/her office, the system assumes that the user does not want to be disturbed. However, no matter how sophisticated the user-aware agents are, they still cannot guess what the user is thinking. Therefore, a meeting in a hallway room represents an informal meeting.

IMPLEMENTATION
To manage such a distributed system, we chose to use a hierarchical multi-agent architecture and to implement the communication between agents with Tcl-dp (a distributed programming extension to tcl-tk implementing RPC).

It has been shown that access control is an aspect of CSCW [16]. As we see on the upper part of the interface, the door icon provides two functions: giving control to the users.

The mirror provides a feedback so that the image on the monitor is the image as seen by the other users. Users often employ the mirror function to center their image.

It is possible to control availability through the door metaphor. Clicking on the door icon makes it vary from an open door (anybody can come and see me), to a semi-closed door (meaning you can glimpse but I don’t want to be disturbed), to a closed door (I'm not available). The door metaphor mechanism is also available in CAVECAT [12].

Ideally, our system wouldn’t need such a control because the system guesses what the user is doing and therefore does not want to be disturbed. However, no matter how sophisticated the user-aware agents are, they still cannot guess what the user is thinking. Therefore, a meeting in a hallway room represents an informal meeting.

Karsenty has taken two meetings, one informal with Madrane and a formal one with Gelin.
TCP/IP. The interimplementation of Tcl-tk and its various extensions allowed us to quickly implement a prototype that is easy to modify. The multi-agent architecture allowed us to easily add/remove user-aware agents and to clearly separate the modules of the system.

In this section we describe the implementation of the logical architecture point of view, then from a physical one.

Agent-based architecture

The architecture is based on a number of agents that collect information about users' activity in order to connect them at the best appropriate time. Given the heterogeneous nature of the agents, it seemed suitable to organize them in a hierarchical way. We thus based our architecture on three kinds of agents: lower-level agents (user-aware agents) that collect raw data, higher-level agents (Intelligent Agents) that analyze the information provided by the user-aware agents, and a server agent (the "brain" of the system) which collects information from the Intelligent Agents in order to make connections between users. We describe this architecture in Figure 3. Information flows from user-aware agents to Intelligent Agents, and from Intelligent Agents to the server.

User-aware agent

We will first define what is a user-aware agent. The idea is to collect as much information as possible about users' physical activity. User-aware agents are software/hardware that provide information about the user's state, e.g. who the user is, what he/she is doing, where he/she is, etc. The way to get such information can be very diverse: infra-red motion detectors, video analysis, pressure captors under the chair, cell phones sensitive to light, badges, keyboards/mouse activity, etc. User-aware agents are low-level agents in the sense that they provide raw data. For instance, they can detect motion, or change of light but do not provide higher-level information such as a user is present or not.

In our case, we decided to use what was available, i.e. the video camera that is part of the mediaspace and the keyboard/mouse activity detectors. Motion detection is done via a simple program that analyzes digitized pictures taken from the camera and considers that motion happened whenever the pixel value between pictures is higher than a given threshold. The keyboard/mouse activity is provided by a Xlib program that intercepts keyboard and mouse events. User-aware agents are not enough to detect accurately whether a user is present or not. Indeed, a user typing/reading is more likely to be detected when not typing/reading than someone sitting idle.

Intelligent agent

The Intelligent Agent is the one that centralizes information from the user-aware agents in order to infer higher-level information. For instance, the previous example of an Intelligent Agent attached to two user-aware agents (keyboard activity, and motion detection) will infer that a given user is present or not, whether this information is provided by the keyboard activity agent or the motion detection agent. IntelligentAgents' knowledge is about a particular physical place which is usually an office. Therefore, they collect user data and don't have any knowledge about the group.

Server

The server has global knowledge of the system. It is the one that CSCW applications will interact with. As an example, if we want to connect user A and B, the server knows which computer they are usually logged on and will send a query to the appropriate Intelligent Agents. Knowing A and B's respect states, the server will guess whether or not to connect them.

Language

The system has been implemented using mainly the interface, communication between processes, and C languages for lower-level routines (image analysis, keyboard activity agent). We show in Figure 4 the main component of the physical architecture and the languages used.

Figure 3: Logical architecture

User-aware agents

Low-level agents that collect raw data about a user's activity, such as motion detectors, video analysis, pressure captors under the chair, cell phones sensitive to light, badges, keyboards/mouse activity, etc. User-aware agents are low-level agents in the sense that they provide raw data. For instance, they can detect motion, or change of light but do not provide higher-level information such as a user is present or not.

Intelligent Agents

Higher-level agents that analyze the information provided by the user-aware agents in order to infer higher-level information. For instance, an Intelligent Agent attached to two user-aware agents (keyboard activity, and motion detection) will infer that a given user is present or not, whether this information is provided by the keyboard activity agent or the motion detection agent. Intelligent Agents' knowledge is about a particular physical place which is usually an office. Therefore, they collect user data and don't have any knowledge about the group.

Server

The server has global knowledge of the system. It is the one that CSCW applications will interact with. As an example, if we want to connect user A and B, the server knows which computer they are usually logged on and will send a query to the appropriate Intelligent Agents. Knowing A and B's respect states, the server will guess whether or not to connect them.

Language

The system has been implemented using mainly the interface, communication between processes, and C languages for lower-level routines (image analysis, keyboard activity agent). We show in Figure 4 the main component of the physical architecture and the languages used.
For each user's workstation three processes were running. The key activity agent implemented in C/X11 scans all the windows and reports when keys are typed, communication is done via tcl-dp in C. The motion detection is implemented in C and uses the Sun Xil library to do real-time motion detection. The algorithm is very basic in order to run fast, pixel difference between successive pictures is calculated, after a set threshold the agent considers that there is motion and communicates the event to the intelligent agent via tcl-dp. The intelligent agent is implemented in Tcl-dp in order to communicate both to the agents and to the server. The server is implemented in Tcl-dp in order to communicate both to the EasyMeeting interface and to the intelligent agents. The server makes calls both to tcl-ffi and to the intelligent agents.

This architecture works well except for a few problems. First, the keyboard activity process must be run by the user otherwise the user must issue the command "xhost +" for the name of the machine running the server. In both cases this is not convenient for the user and the administrator of the system. One solution would be to run the system as root.

Another issue was the number of processes constantly running and slowing down the machine when users are not requesting connections. This problem is solved by using a time-out mechanism which will kill the process if it does not communicate with the server. The solution is to request from the server when a meeting is requested and the server should ask the intelligent agents attached to the user concerned to wake the user-aware agents up in order to get the information.

Finally, we encountered problems due to the not control the physical state of the connections. For instance, when requesting a connection between A and B, A's monitor may switch off and B is not notified. A solution to this problem is to add an agent that will do some image processing to check that an image is not completely black. In this case the problem should be reported to the user.

**DISCUSSION**

In this last section we discuss two issues: the technical and the social, the big brother issue.

What kind of user-aware agents? There is many ways to get information about a user's state. Various hardware/software alternatives can be combined in order to obtain the best results. So far, EasyMeeting uses keyboard/mouse activity and motion detection. More advanced techniques such as face-recognition, which are not very reliable, do not provide real-time recognition. In the following, we discuss different alternatives and their pros and cons.

**Motion detection (video)**

This is a simple efficient captor that can detect motion in real-time. Simple algorithms provide reliable results. However, the result is very limited, we cannot know who is moving, or how many users are moving.

**Face recognition (video)**

Face-recognition is a very active field of research [5, 21], however no real-time efficient algorithm exists for this complex problem. Therefore, we have not used it. Instead, we are currently working on a new approach. Instead of face-recognition, we could use "shirt-recognition" which is much easier. For instance, when the user first logs on the computer in the morning, we could detect the shirt pattern and use it as future reference during the day (hoping the users don't change clothes during the day...). User identification is of crucial importance for the system since this is the feature that allows one to be contacted wherever he is. The most reliable solution is probably badges, however we do not want to impose this on the users.

**Scene analysis (video)**

This is similar to face recognition but more generic. We may want to know, for instance, the user's location in the room, how many users are present, etc. Except for a few simple cases, it cannot be achieved in real-time.
to know the number of users present in the office (i.e., trying to find out if the user is busy holding a meeting), we can tolerate a few minutes delay.

Speech recognition
Such agent has the interesting real-time feature. However, the drawback, is from the human-computer interaction. One has to speak to be identified, which goes against the passive approach of our system. Therefore, such agent could be used as a complement to other captors.

Telephone activity
This captor can easily be implemented and provides very reliable information. In fact, it could be coupled with the previous speech recognition module. Such agent can tell the user is busy on the telephone.

Motion detection
With such information, one can customize the system to tell the user who is calling and who he is talking to.

Software agents
By software agents, we mean the many software that run on a machine which can provide useful information about the user. For instance, if a user runs a calendar manager application, the server can extract information about the user's activities. If a user logs on somebody else's computer, information about the user is also displayed.

Scene analysis
This agent can be used to detect what the user is doing (e.g., whether he is writing at his desk, on the telephone, etc.).

Keyboard/Mouse activity
The simplest form of monitoring is the keyboard/mouse activity. Such activity informs the server that a user is present in front of the computer. However, this agent is however partial: a user might be typing simply reading, and the software agent does not detect activity.

We summarize the pros and cons of the various user-aware agents in Figure 5. We gave five criteria:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Presence</th>
<th>Who</th>
<th>How many</th>
<th>Real-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech recognition</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motion detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Face recognition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scene analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telephone activity</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software activity</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As a conclusion, it is obvious that there is no one all-powerful agent, but each with a specific advantage. A good system will rely on combining the most cleverly many agents in order to build a reliable system.

Big brother issue
A fundamental issue to the whole system is the "Big Brother" effect. Agent will scatter through the office, monitoring the behavior of the users and reminding them of what they have to do. To overcome the big brother issue, we present a number of rules we followed during the design and implementation of the system.

No access to other users' state data
One of the fundamental design of the system is the impossibility for a user to know what another user is doing. When requesting to meet someone, the user cannot be reached. The reason is that it is completely hidden from other users.
Human-computer interaction research is focused on active communications, in which the user, when communicating to the computer, tells what we call passive approach to human-computer interaction, in which the user is not aware of the communication to the computer. On the contrary, in the interactive approach the user is aware of the communication with the computer. In this paper we describe a reactive user interface system that is based on the user-aware agents architecture.

**REFERENCES**


5. Davies, E. and Shepherd, R., "Perceiving and Remembering Faces."

**ACKNOWLEDGEMENTS**

We would like to thank Armand Sani, who greatly contributed to the implementation of the first prototype. Pascale Gros for the exchange of ideas and the helpful discussions. Nabil Madrane for helping implementing the motion detection agent. And finally, many people who participated in testing the system and who gave useful feedback, Jean-Michel Sadoul, Katia Fintzel, Florence Dubois and Khaled Badessaia.

7. Paul Dourish, "Culture and a Convention Space", in Proceedings of ECSCW'93.

8. Fish, R. S., Kraut, R. E., Chalfonte, B. L., "The VideoWindow System in Informal Communications", in Proceedings CSCW'90, Oct. 7-10, 1990, pp. 1-11.


