Easing interaction through user-awarenes

Alain Karsenty

Eurecom Institut

2229, route des Cretes

06904 Sophia-Antipolis

+33 - 93 00 26 63

karsenty@eurecom.fr

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 those agents 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 the user-aware agents are a step toward a better communication man-machine. Instead of the usual approach in which users consciously interact with the machine, we make the computer aware of the users and thus make users unconsciously interact with the computer.

KEYWORDS: groupware, Computer-Supported Cooperative Work, Computer-Human Interaction, Intelligent Agents, Mediaspace.

INTRODUCTION

The keyboard and mouse are today's computers main source of input. Considering the human capabilities and the technology available, this is quite a limitation of what could be done to improve human-computer communication. In this paper, we propose to add as an input source to the computer, the user himself. This is achieved by providing an "eye" and an "ear" to the computer that informs about what is happening at a particular place. As opposed to the traditional (mouse, keyboard devices) and less traditional (multimodal interfaces, speech recognition, gesture based interface input sources), this input is of a passive form since the computer is 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 up information in a file...). 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].

We call the agents that listen to the user's physical activity user-aware agents. To illustrate the use of such agents, we can imagine a simple screen-saver which would benefit from it. Whenever the user is not working in front of the computer, a user-aware agent informs the computer which turns the screen-saver on and locks the screen. As soon as the user is back in front of the computer, the user-aware agent recognizes the user and the screen automaticall unlocks itself. However, this work focuses more 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 distant users. We investigated the use of such agents in a Mediaspace (multimedia system allowing co-workers to communicate through audio/video). In this paper we show how to make meetings between co-workers easier through the concept of user-awareness. For this purpose we developed an application, EasyMeeting, that allows various kind of meetings to be made. Basically, meeting someone through the computer is made as simple as dragging an icon into a window. Once done, the computer will connect the users whenever possible (e.g. they are in front of the computer, and not talking on the telephone, etc

We will first describe EasyMeeting, the application on a mediaspace, the implementation and in particular the architecture, and finally we will discuss the use of agents, and privacy issues.

RELATED WORK

Both the VideoDesk [11] and the DigitalDesk [24] link the real world to the computer world, by enabli to "see" (video analysis). In the VideoDesk, a camera is mounted on top of the physical desk and analyzes the scene. It can for instance recognize the text written on a paper (using Optical Character Recognition (OCR) techniques) and paste it into an application. The VideoDesk recognizes the user's hand movements in order to manipulate objects on the screen. Such work, however, focuses on single-user application and active input, whereas our work focuses on CSCW and passive input.

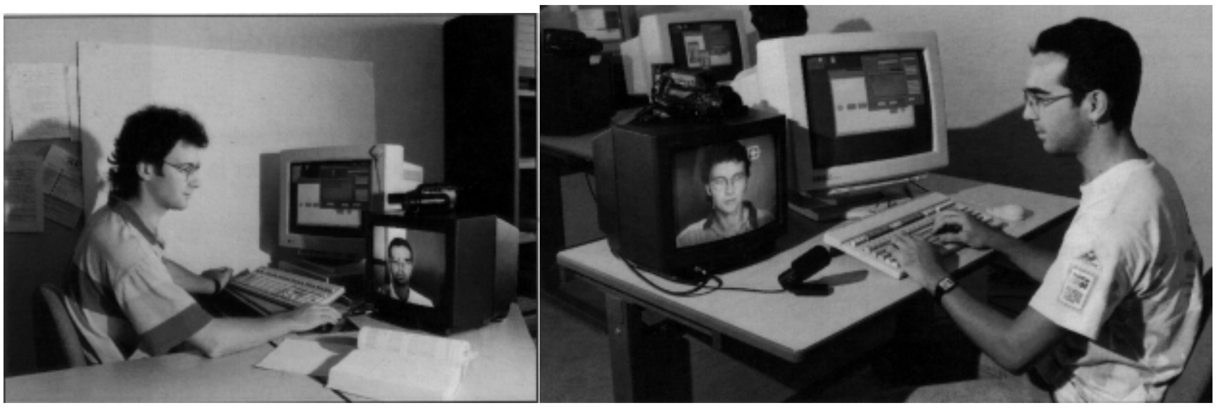
Figure 1 : Connection between two people using the mediaspace.

Multi-agent systems [13, 15] are also 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 [8, 3, 1, 20, 22, 14, 2]. For instance, Portholes [6] uses the mediaspace in an asynchronous way to develop group awareness. Snapshots from people's office 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 differs from EasyMeeting in the sense that it brings to the user the 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 [8] connects two distant rooms through a large video-display, thus improving communication between distant sites.

Finally the concept of active office [10] and reactive environment [4] 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 differs in the sense that we do not want users to wear any device, and we also have all type of agents, not only location ones. In the reactive environment a meeting room is equipped with all kind of sensors to automatically perform the various tasks. The user-aware agents approach is more generic in that it provides a general architecture for such systems and our experience deals more with interaction between users than with room equipment.

EASYMEETING: USER-AWARENESS IN A MEDIASPACE

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

Although reality is not so negative, people in offices still spend a lot of time running after each other. To improve communication, electronic mail is often employed. It's asynchronous, therefore it allows the exchange of information to be made much easier. However, situations need to be discussed in face-to-face meetings, or at least through telephone or video. For this purpose we developed EasyMeeting, an application built on top of a mediaspace that allows to easily meet.

In the next sections, we will first define what is a mediaspace, then we will describe the EasyMeeting application (office metaphor for the user-interfac features...).

Mediaspace

A mediaspace is a system that integrates video/audio and computer networking technology in order to provide a rich cooperative environment. A number of systems have been built at different sites [8, 3, 1, 20, 22, simple systems similar to videophones, which simply connect users through video to more research oriented prototypes. The latter kind looks for more original use of audio/video than simply a telephone with video. Our research focuses on this direction by using user-aware agents to ease the interaction through the

We use an analog mediaspace (Figure 1). It is constitute of nodes connected to a central switch. Each node consists of a monitor, a camera, speakers and a microphone. The switch is computer-controlled through the software Integrated Interactive Intermedia Facility (IIIF) [1]. Moreover, a PIP (Picture In Picture) is connected to the switch, allowing to divide the monitor's screen in four rectangular parts. Through the computer, we can connect until five people together.

EasyMeeting : office metaphor

So far, mediaspaces are accessed through interfaces not so intuitive and simple to use and with limite Shifting from single-user to multi-user app a need to find new metaphors. The user-interface provided (Figure 2) is based on an office metaphor that we believe fits better the needs of CSCW. The same way single-user interfaces are based on the desktop metaphor, it seemed natural to extend it, in the case of CSCW, to an office metaphor. In an office building, we have three basic places 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 rel

From the user-interface point of view, this metaphor is directly represented. Three rooms represent the various level of communication. From formal communication (office room) 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 system makes sure that 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 users explicitly close their access door, this is always possible. The connection only lasts a few seconds.

The user interface has been designed so that meetings are very easy to obtain. Users are represented as icons, when the user wants to meet a certain colleague, he simply drag-and-drops the icon into the appropriate room (optionally he/she can set a later meeting time). Each room represents a certain kind of meeting. Whenever the system thinks it's appropriate, the users are connected through the mediaspace. Furthermore, we provided the ability to create group meetings by dragging icons into an existing user icon.

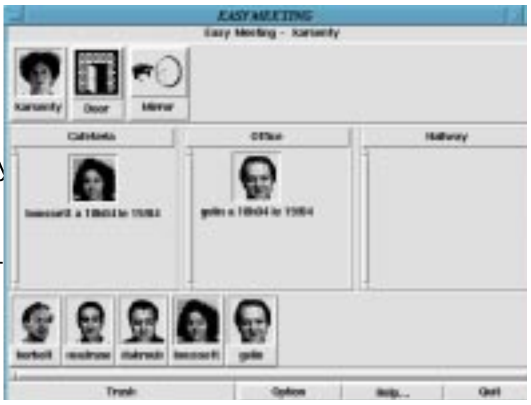

Figure 2: EasyMeeting, user-interface. Karsenty has taken two meetings, one informal with Madrane and a formal one with Gelin.

Access control
It has been shown that access control is a particular aspect of CSCW [16] and mediaspace in particular [7, 17]. As we see on the upper part of the interface, a door icon and a mirror icon provide two functions giving control to the users.

The mirror provides a video feedback. It allows one to see his/her own image on the monitor in order to be aware of the image seen by the other users. Users often employ the mirror function to center their image.

It is also possible to control one's 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 glance in but I don't want to be interrupted), to a closed door (I'm not available e.g. I'm in a meeting and don't want to be disturbed). The door metaphor mechanism is also available in CAVECAT [12].

Ideally our system wouldn't need such a control since the system guesses what the user is doing and therefore what kind of connection he/she wants or doesn't want. For instance, if a user is meeting people at hi, the system detects more than one people and thus 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 someone is thinking. In the previous example, the user might not be in a "serious" formal meeting, and might be willing to be interrupted. Indeed, users need to be provided with some kind of control since all the cases of interaction cannot be taken into account. In fact, the problem of control still has to be studied, and future experiments will help us provide access control features more fitted to the needs.

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 distribute programming extension to tcl-tk implementing RPC and

TCP/IP). The interface was implemented in Tcl-tk. 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 first from the logical architecture point of view, then from 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 suited to organize them in a hierarchical way. We thus based our architecture on three kind of agents: lower-level agents (user-aware agents) that collect raw data, higher-level agents (Intelligent Agents) that analyze the information provided by the various user-aware agents and the server agent (i.e. 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. As we see, information flows from raw data provided by the user-aware agents, to user data provided by the intelligent agent and finally group data stored in the server.

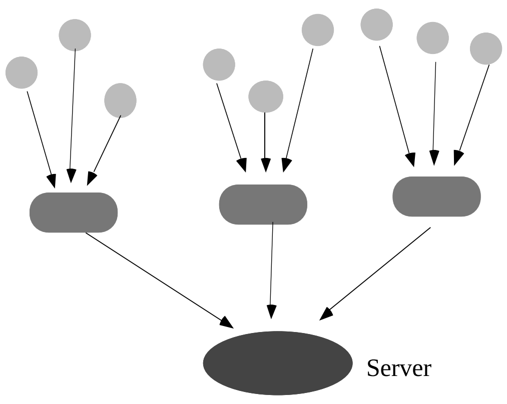

Figure 3: Logical architecture

User-aware agent

We will first define what a user-aware agent is. The idea is to collect as much information as possible about the users' physical activity. User-aware agents are software/hardware that provide this information. Such agents are aware of the user's physical 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 diverse: infra-red motion detectors, video analysis, pressure captors under the chair, cell sensitive to light, badges. User-aware agents are low-level agent 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 or a user is present not.

In our case, we decided to use what we had available, i.e. the video camera that is part of the mediaspace and the keyboard/mouse attached to each user's computer. The information thus provided is motion detection and keyboard/mouse activity. Motion detection is done via a fairly simple program that analyzes the digitized pictures taken from the camera and consider that motion happened whenever the pixel sum value between pictures is higher than a given threshold. The keyboard/mouse activity is done via a Xlib program that intercepts keyboard and mouse events. Those agents are enough to detect fairly accurately whether a user is at his desk or not. Indeed, if a user is typing, he/she is rather motionless, but is still detected. And when not typing text, the user is often in motion (unless reading/writing on the physi

Intelligent agent

The Intelligent Agent is the one that centralizes the information from the user-aware agents in order to infer higher-level information. For instance, in the previous example, 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 are only 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 together, 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 respective status, the server will guess whether or not to connect them.

Language

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

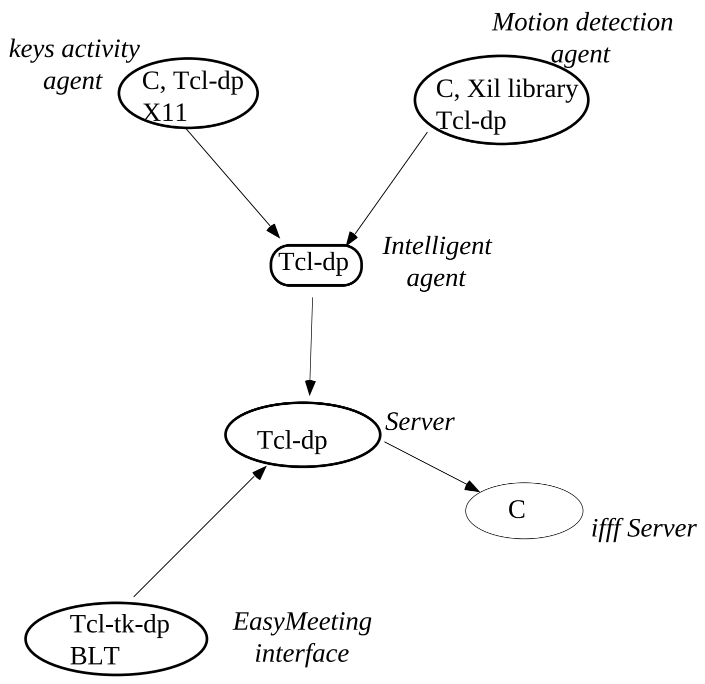

Figure 4: Physical architecture

For each user's workstation we had three process running. The keyboard 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 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 communicate 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 to communicate both to the EasyMeeting interface and to the intelligent agents, and makes call to the iiif server [1] to make the mediaspace connections. Finally, the EasyMeeting interface is implemented in tcl-tk for the graphics, using the tcl extension BLT1.0 to implement the "drag-and-drop" feature and using Tcl-dp to communicate with the server.

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 case this is not convenient for the user and the administrator of the system. This problem could be solved by running the system as root.

Another issue was the number of processes constantly running and slowing down the machines. When users are not requesting connections it is in fact a waste of CPU time to keep the processes running. The solution is to run the processes on request from the server. When a meeting is requested, the server should ask the intelligent agents attached to the users 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 conne
mediaspace. For instance, when requesting a connection
between A and B, A's monitor may switched off, which
will result in a connection where A does not know that B
is connected to him (privacy issue...). A solution is
to add an agent that will do some image processing to
check that images are not completly black, in which case
the problem should be reported to the user.

DISCUSSION

In this last section we discuss two issues, one technical the pros and cons of various user-aware agents, and the second social, the big brother issue.

What kind of user-aware agents?

There is many ways to get information about users' state.
Various hardware/software alternatives can be combined to obtain the best result. So far, EasyMeeting uses keyboard/mouse activity and motion detection. It turned out to be enough for simple cases, when users always log on the same computer and do not want to be contacted in other places. However, one of EasyMeeting's initial goal was to make connections available anywhere in the office. This goal cannot be achieved without some sophisticate
software such as face-recognition, which are not very reliable and do not provide real-time recognition. In the
following, we discuss different alternatives pros and
cons of each one.

Motion detection (video)

This is a simple efficient captors that motion can
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. This is the reason why we haven't used
it. We are currently working on a way aroun
Instead of face-recognition, we could use "shirt-recognit
which is much easier. For instance, when the user first log on the computer in the morning, we could detect the shirt pattern and use it as future reference (hoping during the day the users don't change shirts 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 at this point users to wear any special device.

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 office, how many users are present, etc. Except for a few simple cases, it cannot be achieved in real-time, which might not always be a problem. There are cases when we want 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 side. One has to speak to be identified, which goes against the passive approach of our system. However, such agent could be used as a complement to other captors.

Telephone activity
This captor can easily be implemented and provide very reliable information. In fact, it could also be coupled with the previous speech recognition module. Such agent can tell the user is busy on the telephone, and moreover knows who the user is calling. With such information, the interaction could be customized. For instance, if a co-worker wants to meet user B, and user B is on the telephone talking to a friend, the system could be customized in order to authorize the video connection, despite the conversation on the telephone. Or, user B may customize the system not to be interrupted when talking to customers.

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. Information provided by this agent is however partial: a user might be in front of the computer simply reading, the software agent will not detect activity.

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 from it (e.g. the user is out of the office this afternoon, etc.). The screen lock on signifies that a user is probably not at his desk. If user logs on somebody else's display, his login can be detected making it easy to detect where the user is located. Finally, EasyMeeting itself can provide useful information, since it monitors who's meeting who at different time. From an implementation point of view, the software agents need to communicate with the server, which means to either modify the application (e.g. implement a custom shell, calendar manager, etc.), or, if possible, use existing API (Application Programming Interface) to get the information.

	presence	who	how many	Real-time	what
Speech recognition	*	*		*	
Motion detection	*			*	
Face recognition		*			
Scene analysis			*		*
Telephone activity	*			*	*
Keyboard activity	*			*	
Software activity	*	*		*	*

Figure 5: Pros and cons of various user-aware agents

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

- presence: the agent can detect someone's
- who: the agent is able to identify the u
- how many: detect the number of users pre
- real-time: the agent is able to provide the information in real-time.
- what: the agent is able to detect what t user is doing (e.g. the user is writing at his desk, on the telephone, etc.)

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

Big brother issue
A fundamental issue to the whole system, is the "Big Brother" issue. Agents scattered through the office, monitoring the actions of the users reminds too much of a telesurveillance system... We ought to design a system that users will trust. Nobody will use a system where it is possible to spy on everyone's actions.

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 a connection with someone, if the user cannot be reached, the reason why is completely hidden to other users. If one wants to find out the reason

he/she cannot get in touch with someone, traditional methods needs to be employed. For instance, one should ask the secretary if the college is out of the office, or go physically to the college's office.

Our approach is different from other systems such as Portholes. They aim at developing group awareness, a sense of co-presence, by explicitly graphically making users co-present, whereas our aim, in case of EasyMeeting, is to ease the interaction through the mediaspace. Therefore, privacy can and should be respected.

Evanescent data

This is the computer aspect of privacy. Data, such as who is where, doing what, should not be stored unless necessary. For instance, when a connection is requested, the server will simply ask the various agents to find the user. Whenever the connection is established, there is no need to keep the information about the user.

By doing so, we ensure that in the design of the system itself, there is no way to break privacy rules. We therefore make this technology more trustable.

Ubiquitous/invisible agents

From a hardware point of view, the agents can be numerous and scattered through the building. In order to make the users not feel "watched", ubiquitous agents should be made part of the environment, thus invisible. In that sense we move toward Weiser's view of ubiquitous computing [23].

In many mediaspaces, including ours, a node consists of a monitor, on top of it a video camera, a microphone on the table or attached to the camera, and next to it the workstation. The whole system takes up a lot of physical space, and is far from being invisible... Systems such as Hydra[18] provide interesting alternatives to be used in mediaspaces: a hydra unit consists of a small box made of a small camera and a mini-display, which can be easily moved around. A meeting consists in using many hydra units next to each other.

WISYYSM

We can derive from the acronym WYSIWIS (What You See Is What I See) often used in groupware [19] the acronym WISYYSM (When I See You, You See Me). One way to apply it is to only allow two-way connections. One-way connections, which would allow someone to spy in somebody else's office, are thus not possible. However the WISYYSM can be extended in many other way. For instance, when someone glances in a college's office, a snapshot may be taken. In a similar fashion, an agent could detect when someone physically glances in a college's office and take a snapshot of the user peeking through the door. When the college comes back to the office, snapshots can be replayed, thus the college can find out who is trying to reach him/her. This way we also respect privacy, since a user cannot use the glance feature to "spy" in somebody else's office.

CONCLUSION

Human-computer interaction research is focused on active mode of communications, in which the user is aware when communicating to the computer. In this paper we describe what we call a passive approach to human-computer interaction, in which the user is not aware to communicate to the computer. On the contrary, it is the computer that is made aware of the user. On this principle, we have described a user-aware agents based architecture for CSCW and an application to the mediaspace, EasyM

This work is continuing in different direct

First, we need to further experiment. This is still a prototype, and we are currently making it more robust. Experimentation will allow us to validate the ideas s discussed in this paper.

Second, we plan to add more adaptors than keyboard/mouse activity and motion detection. The most important one is user identification. Even if face recognition is not reliable yet, it will still be useful to do some bas

Finally, we are interested to apply the user-aware agents architecture to other applications, CSCW or single-user. Many applications can take advantage of this concept (e.g. the screen-saver described in the introduction, teleteaching, shared editing, etc.) and this will also help us refine our architecture.

ACKNOWLEDGEMENTS

We would like to thank Armand Sanier who greatly contributed to the implementation of the first prototype. Pascal Gros for the exchange of toughts and the help in setting up the mediaspace. Bernard Merialdo for the useful discussion. Nabil Madrane for helping implementing the motion detection agent. And finally the many people who participated in testing the system and gave useful feedback, Jean-Michel Sadoul, Katia Fintzel, Florence Dubois and Khaled Bousetta.

REFERENCES

1. Buxton, W. and Moran, T., "Europarc Integrated Interactive Intermedia Facility (IIIF): Early Experiences," in Multi-User Interfaces and Applications, S. Gibbs and A. A. Verrijn-Stuart, Eds. North Holland, 1990, pp. 181-188.

2. Buxton, W., "The three mirrors of interaction: A holistic approach to user interfaces." In L. W. MacDonald & J. Vince, editors, Interacti Environments. Wiley: NY, 1994.

3. Sarah A. BLY, Steve R. HARRISON, and Susan IRWIN, "Media Spaces: Bringing People Together in a Video, Audio and Computing Environment", Communications of the ACM, 36(1), p.28-47, January 1993

4. Cooperstock, J. R., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W., "Evolution of a Reactive Environment", In Communications of CHI'9

5. Davies, E. and Shepherd (eds), "Perceiving and Remembering Faces", Academic Press, Lond

6. Paul Dourish, Annette Adler, Victoria Bellotti, and Austin Henderson, "Your Place or Mine? Learning from Long-Term Use of Video Communication", Europarc, Technical Report EPC-94-105, EuroPARC, Cambridge 1994.

7. Paul Dourish, "Culture and Control in a Media 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.

9. Grudin, J., "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces," in Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'88), Portland, OR, September 26-29, 1988, pp. 85-93.

10. Harter, A, Hopper, A, "A Distributed Location System for the Active Office", In IEEE Network, vol. 8, no 1, pp. 62-70, Jan/Feb 1994..

11. Krueger, Myron, W. Artificial Reality. Reading: Addison-Wesley, 1993.

12. Mantei, M., Baecker, R., Sellen, A., Buxton, W., and Milligan, T., "Experience in the Use of a Media Space," in CHI'95, 1991, pp. 203-208.

13. Martial, F., V., "Coordinating plans of autonomous agents. LNAI 610, Springer-Verlag: Berlin.

14. Margrethe H. Olson and Sara A. Bly. "The Portland experience: A report on a distributed research group." International Journal of Man Machine Studies, 34(2):211-228, 1991.

15. Rizzo, P., Cesta, A., and Miceli, M., "On Helping Behavior in Cooperative Environments", in International Workshop on the Design of Cooperative Systems, June 1995.

16. Tom Rodden, Gordon Blair, "CSCW and Distributed Systems: The Problem of Control" in Proceedings of the Second European Conference on Computer Supported Cooperative-Work (ECSCW'91), September 25-27, 1991, Amsterdam.

17. Salber, D. and Coutaz, J., "Fenetres sur groupes des mediaspaces pour collaborer et communiquer." In Proceedings of Interface des mondes reels et virtuels, Montpellier, pp. 309-318, Feb. 1994.

18. Sellen, A., Buxton, W. and Arnott, J. "Using spatial cues to improve videoconferencing" in Proceedings of CHI'92, pp. 651-652, 1992.

19. Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., and Tatar, D., "WYSIWIS Revisited: Early Experiences with Multiuser Interfaces," ACM Trans. on Office Information Systems, vol. 5, no. 2, 147D167, April 1987.

20. Thomas, R. H., Forsdick, H. C., Crowley, T. R., Schaaf, R. W., Tomlinson, R. S., Travers, V. M., Robertson, and G. G. "Diamond: A multimedia message system built on a distributed architecture." IEEE Computer, volume 18, pages 65-78. 1985. Reprinted in Greif, 1988.

21. Turk, M. A. and Pentland, A. P., "Face Recognition Using Eigenfaces" in Proceedings IEEE, 19

22. Kazuo Watabe, Shiro Sakata, Kazutoshi Maeno, Hideyuki Fukuoka, and Toyoko Ohmori. "Distribute multiparty desktop conferencing system: Mermaid." pages 27-38. ACM Press, New York, NY, October 1990.

23. Weiser, M., "The computer for the 21st century." Scientific American, pp. 66-75, Sept. 19

24. Wellner, P.. "Interacting with paper on the DigitalDesk." Communications of the ACM, July 1993, vol. 36, no 7, pp. 87-97.