Chapter 3

Environments for Cooperating Agents: Designing the Interface as Medium

A. Dix, J. Finlay and J. Hassell

3.1 Introduction

Various interface styles suggest paradigms for understanding interaction. Direct manipulation (OM) suggests the interface as a passive entity, providing tools for the user to control. Intelligent interfaces suggest instead an active interface, a colleague which (or even who) cooperates with the user on the task in hand. Each of these paradigms seems useful in different contexts. Matters become more complex when we consider systems with multiple applications or multiple users. We can no longer see the interface as part of a two-participant dialogue, involving human and computer. Instead, we look towards an environment where several active participants - some human, some automatic - cooperate.

In this chapter we propose that viewing the interface as a *medium* allows us to make sense of the interplay between passive and active components of an interface, and, indeed, of that between human users. Within an interface we will distinguish the *objects,* the passive elements; the *agents,* the active; and, most importantly, the *medium,* the environment within which agents act upon the objects and communicate with one another. We consider how this model can support our understanding of the interaction, taking examples from intelligent interface design and Computer Supported Cooperative Work (CSCW). Consequently, we must design the interface as a medium of communication: an environment in which both human and artificial agents can cooperate effectively (for further discussion of cooperation, see Chapters 1, 2, 5, 6, 9, 10 and 11).

3.2 History

The analysis of an interface as composed of *Agents,* the *Medium* and *Objects* (AMO) was proposed by the authors some years ago as a way of understanding the interplay of passive and active elements within single-user interactive systems (Dix and Finlay 1989). Of particular importance is the medium, the environment within which the user interacts with the objects and other agents, human or automated, in the system. The application of AMO to single-user systems made use of images drawn from day-to-day interpersonal communication to understand the human- computer dialogue. We now bring the approach full circle by focusing again on multi-user and multi-agent interfaces. The chapter is based partly on older, but previously unpublished, material and partly on more recent implementation and analytic work. Several ideas which seemed "off the wall" when we originally discussed the AMO model are now only relatively simple extensions of current systems and metaphors. However, although concepts such as identifiable interface agents are now part of the normal vocabulary of Human-Computer Interaction (HCI), the medium itself is not. But it is through the medium that human users cooperate with artificial agents and with each other, and thus our emphasis must shift towards the positive and explicit design of the medium itself.

We begin in Section 3.3 by discussing the background of active and passive paradigms in single-user interaction. This is used as a springboard for the discussion of the AMO model in Section 3.4. The model is used to discuss design issues for adaptive interfaces (Section 3.5), concluding that the agent of adaptivity should be embodied in some form within the interface. In Section 3.6 we discuss an experimental system which exemplifies this principle of embodiment: an adaptive "buttons" orientated interface. We then shift our attention, in Section 3.7, from artificial agents to other people. We see how a medium-orientated perspective casts light on some design issues in electronic conferencing and communication. Finally, in Section 3.8 we give some suggested design heuristics for the medium, incorporating both other agents and other users. However, this discussion is not intended to be complete; the suggestions are, we hope, useful, but not crucial. The primary goal is to establish the central importance of the concept of, and design of, the interface as a medium.

3.3 Active and Passive Interfaces

Different interaction styles suggest different paradigms for understanding interaction. DM interfaces emphasize the passivity of the interface: the user is interacting with *things* in an artificial world (for example, on a desktop). This interface style is highly popular, partly because of the naturalness of the physical metaphors used, partly because of the immediacy of response (if you want something to be done you do it, rather then telling the system to do it). In addition, the very passivity of the system gives the user a sense of control; the initiative lies with the user. There is a danger that such user-controlled dialogues will be *under-determined* (Thimbleby 1990), but this is largely obviated by the graphical presentation of the objects of interest.

Successful as such interfaces are, the concept becomes dangerously stretched when extended beyond those applications which are most well suited, such as drawing and simple word processing. In an application such as statistical analysis you clearly want the machine to *do* something *for* you rather than you doing it for yourself. DM is an excellent paradigm for the production of tables of data, but when faced with analysis, especially non-standard analyses, the limitations become obvious. DM techniques can be used to draw a diagram describing the statistical processes required, and enabling intermediate results to be seen, so encompassing many of the positive points of DM. However, in the end, you want the machine to do the actual calculations: after all that's why you're using it.

Even the classic what-you-see-is -what-you-get (WYSIWYG) word processor starts to fail when more complex facilities are demanded of it: style sheets are added to paragraphs, and alternative views may even be given, containing, essentially, text formatting languages. Basically, when the issue is simple page layout or simulated typewriting then DM is sufficient. However, as soon as the focus changes to document processing then issues such as consistency of style make us demand that (again) the computer works for us.

Intelligent interfaces, on the other hand, emphasize the active nature of the interface. The interface sits between the user and the application, and uses its knowledge of system semantics and (possibly) of user understanding to present the application to the user in what it deems is an appropriate fashion. Adaptivity may identify the user's understanding of a topic in order to provide a restricted functionality (cf. Training Wheels, Carroll and Carrithers 1984) or alter the presentation of the system's functionality according to commonly used commands (Mason and Thomas 1984). Alternatively, an adaptive system may alter the view of the domain in order to reflect the user's actions (Greenberg and Witten 1985). Regardless of the aspect of the interaction which is adapted, there is a danger that the user will suffer due to loss of control over the interface and a sense of instability.

Traditional User Interface Management Systems (UIMS) fall between these two paradigms. They act as a bridge between the user and some set of underlying functionality (of which the statistical system would be a good example). Applications know little about the specific user interface and are addressed purely at the application domain. The job of the UIMS is then to produce a means of accessing the functionality, and presenting and manipulating the objects in the application domain. This philosophy of interface independent applications (and even sometimes application independent interfaces!), and the language models that underlie it, has been the subject of an extensive critique, which more modern UIMS are attempting to address. In particular, many try to be more "OM-ish", but of course they are addressing the application areas where OM has trouble.

One issue that has been a problem with older UIMS is that the UIMS's knowledge of the application was restricted to a type-syntactic description of function calls available. In order to produce sensible interfaces, deeper semantic knowledge is often needed. To address this, several recent UIMS proposals include knowledge bases and expert systems components which overlap to some extent with the intelligent and adaptive interfaces.

In both the traditional UIMS and the intelligent interface the user interface is seen as a *mediator* between the user and application. The user communicates intentions to the interface agent, which then processes these and passes them on to the application. In order to obtain reasonable performance, many UIMS allow direct semantic feedback - the user talks directly to the application - largely obviating the original intentions of separation. In each case we have a three-agent communication.

3.4 The Interface as Medium

In Section 3.3 we saw two very conflicting views of the interface. One, OM, emphasizes the system's passivity, but has difficulty coping with active applications. The other, represented by both the adaptive interface and the UIMS, has a far more active interface component, but lacks that feeling of directness and control that makes OM so popular, especially among the computer naive.

We take a third view of the interface, as a *medium,* which allows us to make sense of the interplay between the passive and active components of an interaction. The word "medium" here is taken to include the whole software/hardware amalgam, with both its functional and aesthetic attributes. In particular, it is not limited to the information theoretic concept of a channel or the physical characteristics of a device, although these will both be facets of the medium. In this paradigm, we decompose a system into *agents* (human or machine), the *medium* and *objects* (Dix and Finlay 1989).

Objects here are not those of an object-oriented system; only the agents are capable of autonomous action. The image is of a physical object: it doesn't *do* anything but it is manipulated by the various agents in the environment. The term "medium" is drawn wider than its use when we say that oils are an artistic medium. We would also call a style, such as Cubism, a medium encompassing a whole set of conventions and constraints. Similarly, television as a medium includes not only the technical limitations but also the cultural, legal and economic constraints. The medium through which you communicate, and in which you operate, determines the way in which you frame your actions and interpret your perceptions. In our context the medium is the environment through which agents interact with one another and with objects. We are interested in both qualitative and quantitative attributes of this medium.

A motivating example for this approach is traditional mail and communication systems. Clearly when viewed as theoretical communication channels most such systems are similar: they differ more in the qualitative aspects of the interface. The important thing to note is that these non-functional differences, such as pace and ease of interaction, can make profound differences to the content of communication. This is typified by the differences between face-to-face, telephone and paper mail communication.

In the mail example, the total system consists of the medium and the people who are communicating. The medium is (relatively) passive and the people active. In general, systems have active members other than the humans and we refer to both types of active member as *agents.* The final classification, *objects,* refers to those components that are passive, but not merely artefacts of the interface, for instance data files. As with many such classifications it can be carried out at various levels in the system description. For example, at one level the postal system is a medium through which agents communicate, whereas, at another level, it involves the letters as objects being manipulated by the postman, an agent. Another example is a washing machine: this is an object when it is being lifted out of a van, but an agent when washing clothes.

When we first used the AMO model, the principal agents within a system were the applications and perhaps other users. The term was a way of identifying the active parts within systems, but was not normally used by the systems themselves. Now, of course, the idea of agents has become commonplace, but it is still important to look for interface elements that are not designated "agents" by the developer and yet exhibit autonomy - we will see examples of this later.

It is also important to note that agents within the system may have different levels of autonomy and intelligence. An automatic mail reply tool is an agent which acts autonomously, but without intelligence. An adaptive macro generator acts with some intelligence. Human agents obviously are fully autonomous and intelligent!

3.5 Adaptive Interfaces

The distinctions introduced can be used purely descriptively or normatively in judging existing or putative systems. An example of an interface issue that can be addressed is that of adaptive or intelligent interfaces (see also Chapter 10). These are sometimes justified by analogy with human dialogue (Kass and Finin 1988) . When we enter a dialogue with. other human agents we expect them. .to. adapt the level and style of the dialogue to their perception of our abilities, knowledge and. alms; the same type of adaptive dialogue, it is argued, should be possible with computers. On the other hand, such adaptive systems may be unpredictable to use, leaving the users feeling lacking in control, as the system continually tries to second guess them, and unsure of the response from (and even the method of achieving) system operations. Again, we see the conflict between a user-controlled passive (but stupid) interface and a more active, intelligent, and independent one. Can the AMO distinctions help us to resolve this conflict?

By analysing appropriate real-world situations in terms of agents and media, we can make recommendations as to which parts of a computer system should be subject to adaptivity. If we examine again the analogous human- human dialogue, it consists of two agents, the participants, and the medium through which they communicate. In the case of face -to-face conversation the medium would include the air through which the sound waves travel, the language used, the non-verbal visual cues, and, in a wider setting, the room in which the conversation is taking place. The important thing to note is that it is the other participant (the agent) who adapts and the medium which remains relatively stable. We would find it extremely disconcerting if the air around us began to vary its properties in sympathy with its model of us, perhaps reducing the speed of sound to a few centimetres per second in order to slow the rate of dialogue down, or adding echoes if it thought we were missing things. The closest effect one could imagine would be chatting next to a sound sensitive disco light controller!

Of course, changes of medium do occur. However, these tend to occur at a low rate and/or under the mutual agreement of the parties. For instance, if telephone conversations are interspersed with written letters, the change in pace and context of the medium is rather obvious. One can consider more dynamic situations, such as a technical "brainstorming", which swap between use of direct speech, whiteboard, and pencil and paper. However, in this case, the alternative media are supplementary rather than representing a total change in the medium of dialogue.

If we look again at adaptive interfaces, the situation is similar to human- human dialogue - we have two agents, the user and the application, and the interface, which is the medium between them. However, in this case, it is not the other agent (the application) which adapts but the medium itself. That is, the supporting analogy breaks down on who and what is adapting. It is not surprising, then, that the user may feel out of control, as the very means of communicating with the system is neither predictable nor stable. Users must possess such a *deterministic ground* in order to have a context from which to view and interact with the less predictable parts of a system (Dix 1990, 1991). A possible solution is to introduce an active agent into the system which will cooperate with the user on the task in hand. The adaptivity is thereby shifted from the medium (the interface) to another agent. The interface retains its consistency and the adaptivity is localized.

We can think of this approach as *embodied adaptivity.* There are two ways in which adaptivity can be embodied in the interface. The first is if the adaptive agent itself is given an embodiment. Hewlett-Packard's New Wave agents are an example of autonomous agents, embodied in an icon in the interface. One can imagine similar agents given an element of intelligence and thus being a focus for interface adaptivity. An example of an existing system where the adaptivity is given such an iconic embodiment is EAGER (Cypher 1991). This system looks for commonly occurring command sequences and when it thinks that it can predict your future commands, a smiling cat appears, highlighting its suggestion for your next command (menu selection etc.). This can either be ignored or the prediction confirmed.

The second way of embodying adaptivity is to embody the *results* of adaptivity. In the Xerox Buttons environment snippets of Lisp code, which act on the user's electronic environment, are attached to on-screen buttons (MacLean et a1. 1990). When the buttons are clicked, the Lisp code is executed on the user's behalf, a form of sophisticated macro. Buttons can be copied and amended by the user, making them a focus for the user's own *adaptation* of the interface. However, although the buttons are autonomous, in that they perform tasks *for* the user, they are not intelligent, and are thus adaptable but not adaptive. Similar facilities are now becoming available on commercial word processors and spreadsheets, but with more limited macro languages, and tied to particular applications. As well as being more flexible, the buttons are *first class,* that is, they are manipulable items in the interface. In particular, they can be mailed (locally) from person to person. This happens frequently because the users who require adaptation may not always be sufficiently proficient Lisp programmers, so one user may produce a button and then give it t6 another user. Thus, the button has also become a focus of adaptivity within the social group. The agent of adaptivity is the colleague, a human agent. It is only a small step to consider using the non-intelligent button as the embodiment of the results of an adaptive interface. This is precisely the approach taken in the experimental system described in Section 3.6.

3.6 An Experiment in Embodied Adaptivity

Current work in user modelling for adaptivity has mostly been restricted to the application level, despite the fact that much of the information contained in the model has a wider applicability than the scope of the application domain in question. It seems a waste that this domain-independent information, which has been acquired at some considerable cost, is only allowed to affect the operation of the application in which its model resides.

The different levels of generality of the information stored in the model seem to indicate that the information should be stored in a hierarchical structure with the more application-independent information further up the hierarchy and the more application-specific information being stored closer to the leaves (which in this case could contain information relating to the current instance of the application usage, for example, information specific to the current document being worked upon).

This hierarchical structure provides an efficient way of storing user modelling characteristics and also allows for inheritance of general characteristics by any new tools introduced to the system. The reverse operation - of specific information propagating up the hierarchy as it is inferred to be more general than the level of the hierarchy at which it resides - also aids the process of task recognition in a global system-wide context.

The advantages of this structuring are obvious. With current advances in multi-window, multi-tasking operating systems, people are adopting a more concurrent tool-based approach to work. The boundaries between applications are no longer rigid, and interfaces are becoming more data-orientated rather than application-orientated. These new work styles require adaptation to break out of its application-specific role and to have effect on the system as a whole if it is to be useful in any real sense. The hierarchical structuring of the information parallels closely the way in which tasks are repeatedly broken down into subtasks, until the subtasks can be achieved by an application.

This approach to whole-system modelling allows any agent, whose job is to perform adaptation, to range over the entire user environment (the medium), thus giving it scope to provide a consistent adaptive interface to the system as a whole.

As an initial trial of the ideas presented here, an experimental system has been designed. The domain chosen for the prototype is that of document processing under LaTex on UNIX graphic workstations. There were a number of reasons for this choice: firstly, this is the environment that a good number of the researchers at York use for their document processing requirements; secondly, the production of documents using the system requires the use of many diverse tools, and the set of these and the environment in which they are used has been shown through questionnaires to be a matter of personal preference; and finally the complexity of mastering the tools, methodologies and command syntax is substantial.

The concept of the system is essentially simple - a button-dock is introduced to the machine's work surface with buttons that automate the user's habitual tasks (for example, there may be buttons for *print, spell-check* and *preview).* These buttons can be seen as intelligent agents working in the environment. Buttons are created or destroyed automatically by an adaptive filter working on a trace of the user's interaction with the command line and also with the buttons themselves (the collection of which is transparent to the user).

The filter builds up a hierarchical model of the user's interaction, starting at the lowest level of the tree - the specifics of the document in question. Commands that are used habitually are migrated to a button with a script specific to the current document. Over time, inferencing techniques within the agent recognize similarities between the scripts of buttons at the same level in the hierarchy and automatically form a more general script at a level one higher than the specific scripts. Any new document then inherits the general buttons of the levels above it to give a default button-dock for the new instance. As buttons are created automatically, so their traced use is also subject to adaptation, and buttons that are not used will gradually be destroyed and replaced by others.

To ensure stability of the interface, and to comply with the argument that the medium should not be subject to adaptation, the addition of the buttons to the interface does not in any way change the methods with which the user can get the job done. The buttons are a simple adaptive addition to the interface, the introduction of new agents, and can be completely ignored by the user if he or she so wishes.

The application of the AMO paradigm to this area provides important design heuristics to constrain the adaptivity in the system when the agents in question are autonomous adaptive agents. In Section 3.7 we change our focus to consider how the paradigm can be used to analyse human- human communication in computer supported conferencing systems.

3.7 Conferences and Cooperation

Imagine a physical conference room. What is it like? Typically, you would expect to see chairs and tables, an overhead projector, a whiteboard, a screen, and, if it is in use, people with many bits of paper. They do not constitute a conference in the same way that, say, a collection of pipes, boilers and turbines might constitute a power plant. Instead, the conference room is a medium in which the participants are able to make for themselves a conference. Often, the basic geometry of the room is varied from meeting to meeting, some groups opting for a linear "boardroom" effect, others placing chairs and tables in a circle, others stacking the tables and having ranks of chairs looking forward at the speaker. It is usually the case that such furniture shuffling takes place prior to the meeting proper. That is, the meeting has two phases: in the first the room is the medium and the chairs and tables are objects to be manipulated; in the second, the conference proper, the geometry becomes part of the medium itself.

In special-purpose rooms, such as lecture theatres or boardrooms, where the range of use is well understood, the geometry may be fixed. This usually renders such rooms virtually unusable for other purposes. This has been recognized in many areas, for example in church interiors, where the fixed, forward facing Victorian pew is now often replaced by movable chairs to allow different geometries for Sunday worship and for alternative weekday activities.

Occasionally technical constraints may force the room's designer to fix the geometry. This was the case with the design of Capture Lab (Mantei 1988), a computer assisted meeting room. Mantei describes how difficult this design process was, and the many different design options that were considered. However, the eventual design, the product of much careful thought, still conflicted with the participants' social expectations - the participants had to learn the new positions of power and cooperation that went with the new environment. To some extent this re-orientation was an inevitable consequence of new technology, but it underlines the complexities of designing a system for conferencing, even where the main elements are physical and well understood.

A medium-orientated approach to electronic conferencing would suggest that the emphasis should be on producing a medium for communication, which the participants should shape to their needs, like the conference room with chairs and tables; that is, a collection of shared items and tools that the participants can structure to their own needs. One would not dream of designing a physical conferencing room around a model of a typical meeting agenda. However, that is precisely the approach of structured communication systems such as Coordinator (Flores et al. 1988). More in line with the medium-orientated approach would be locally structured conferencing systems such as the Amsterdam Conversation Environment (Dykstra and Carasik 1991), where the system has minimal structure, allowing the participants to build the conversation or conference of their choice.

On the whole, conferencing and communications systems, even the most loosely structured, stand apart from normal single-user interaction. However, the participants in the physical room would bring in objects from other work contexts: briefcases, piles of papers, overhead acetates. Similarly, we should eventually look towards a communication medium which is seamless with the normal single-user medium, rather than a conferencing "application" separate from the rest of the interface. The closest extant system to this ideal is probably the Doors metaphor (Cook and Birch 1991), which is similar to Rooms (Henderson and Card 1986). As in the Rooms metaphor, the user can move between several virtual rooms, but, unlike the original metaphor, some of the rooms are private works paces - normal electronic desktops - and some are shared. The user can carry objects back and forth between these private and shared rooms. A more mundane, but more prevalent, example is the growing sophistication of "enclosures" mechanisms in commercial LAN-based email systems.

3.8 Designing the Medium

We have seen how important it is to distinguish the adaptive agents from the medium within an interface, and discussed a medium-orientated approach to electronic communication. We now address the more general question of how to design the interface as a medium within which the user can cooperate with other users and with electronic agents in a unified manner. The medium should be invisible to the users themselves: it is the environment within which they work - natural and transparent. But to obtain that effect it must be highly visible to the designer. All interfaces constitute a medium - the issue for the designer is not whether to design a medium, but whether the design should be implicit, an accident, or whether to make the design of the medium explicit.

In many circumstances, the medium is largely "given", part of the operating system or window manager. Developers can build upon this given medium, as an artist adopts a particular style in the use of oils, but they operate under strong constraints - the effect on the medium of these pervasive, underlying systems is enormous. However, these pervasive media, designed for single-user/single-application interaction, will become increasingly strained as integration between users and between applications increases, and autonomous agents (surrogates, filters, helpers) become commonplace. To encompass these diverse developments we need a fresh look at the design of these underlying systems and the media they portray.

We do not attempt a complete answer to the difficult problem of designing the medium, but suggest a few useful heuristics and directions: active/passive separation, equal access, use of existing multi-user and of single-application media. However, the important issue is that the medium *needs* to be designed.

3.8.1 Separation

One issue that has already arisen is the separation of the active and passive parts of a system. It is easy to let active functionality "creep" into those parts that ought to form the user's deterministic ground. This is frequently a problem in modern hypertext systems. Early descriptions of hypertext gave the impression of a passive system, a sort of highly connected electronic book. This is still the impression given by, say, address book applications and is largely the model portrayed to the hypertext viewer. However, in many authoring systems the implementation is very different. Each button on a card triggers a piece of code, one possible action of which is to move to another card. This allows more complex systems than the simple book metaphor would suggest.

For an example of this problem consider a tourist information system. It has one card in which hotels are listed alphabetically, but the user can also look at an area map and then navigate to a list of all the hotels in that area. If you take a passive model of the hypertext, then you know that every time you return to the hotel listing the same information will be displayed, but you have no guarantee (except the accuracy of the data input) that the alphabetic and the area listings are consistent. An active view, where the lists are seen as computed, does guarantee consistency between views, but not necessarily through time - we all know systems where each time we ask the same thing we get a different answer. Which model should the user adopt? The clues as to which is the appropriate model are often absent and yet the models make very different predictions about the system. If the developer is careful, a hypertext need not suffer this problem, but it is an easy trap to fall into.

Designers, whether of hypertext or any other interface, should be clear in their own minds as to which elements of the interface are active and which passive. This should be communicated to the users, by documentation and by clues in the presentation as to the affordances of the objects.

3.8.2 Access

In real life, through social and physical constraints, we do not have equal access to all objects. However, when cooperating closely with others it is often the case that the principal objects and the medium of communication itself are equally available to all. This is close to Thimbleby's principle of *Equal Opportunity* which suggests (among other things) a blurring between the items in the interface used for input and output (Thimbleby 1986,1990). If we design the interface as a medium, we expect it to behave in roughly similar ways when sending information to and from the user. However, the facilities normally available to the user and to the application are very different. Happily, there are some exceptions.

Took's Presenter system (Took 1990) treats the interface graphics as a data structure. Items may be moved, resized and edited, by the application and by the user. This is not an anarchic situation as the movability constraints and the groupings of items can be set so as to mimic standard interface widgets (and novel ones). However, the overall impression is that the surface interface is an area of negotiation between the user and application, a medium of communication.

Another example of good practice is the HyperCard *development* environment (Apple 1987). If the programmer wants a script to draw a filled circle, then the script selects the circle from the toolbox, clicks at one corner, clicks at the other (drawing the outline), then selects the paint can from the toolbox and clicks in the middle of the circle (filling it). The script performs exactly the same actions as the developer would to paint the circle by hand. The card surface is the common object upon which the programmer and the programmer's scripts are working, using the shared medium of the toolbox. This, of course, is the situation for the HyperCard *programmer;* the user may, depending on the skills of the programmer, suffer exactly the paradigm problems described earlier.

3.8.3 Using Models of Interpersonal Communication

One way to design the medium is to take an existing electronic human–human communication mechanism and use this for communication with other non-human agents. Email systems are an obvious candidate, and, indeed, it is possible to send email messages to special mailboxes which act as "servers", perhaps mailing back a document, or adding you to a distribution list. Structured mail messages, such as those in LENS (Malone et al. 1986), offer more opportunities for communication with other agents. Indeed, the Mailtrays system (Rodden and Sommerville 1991) is described in terms of "a federation of cooperating, distributed agents" which communicate using structured email - the medium. For example, to compile a program, one mails a "compile" form to a compiler agent, which returns a "report" form when the compilation is (successfully or otherwise) complete.

To return, yet again, to hypertext, this is used as the object of cooperation or the medium of communication in several conferencing and shared editing systems (e.g. ACE or Quilt). The hypertext in such systems is clearly passive - the agents of change are other participants. Social protocols, locking schemes or automatic change notifications are used to inform the participants of one another's actions. This suggests that hypertext may be used as a medium itself between user and intelligent agents. This gives a possible way forward for the separation problem in hypertext: the text itself is passive, but it is altered in specific places and times by a cooperating agent.

3.8.4 Using Models from Application Interfaces

One can also look at application interface paradigms and ask whether these can generalize to ways of communication with other users, or multiple agents. In the earliest interfaces, the interaction was restricted to a single application. This single focus has been developed in two quite different directions in more modern multi-window graphical interfaces.

One paradigm is seen in Macintosh and Next systems: the screen has windows representing objects from different applications, but only one application is active. There is an implicit address to the user's actions, the current application. One can view this in two ways, either as looking through the application to the desktop (as suggested by the Macintosh's menu border), or as looking at the windows, with the application by one's side. This "over the shoulder" view of the application would suggest multi-user interfaces of the "shared window" or "shared screen" type.

The other paradigm is exemplified by the many UNIX window systems. In these, the application is embodied within a window. There is still a single active application, but this is *within* the active window. The visual clues which emphasize this are the fact that menu bars and status information tend to be placed within the windows, as opposed to on the desktop. The interpersonal communication suggested by this paradigm would be where a user was embodied within a window, basically a "phone" type connection.

Both paradigms have the data objects within the applications. This is less clear in the Macintosh type interface, but true of both. Indeed, this is how applications are traditionally coded, the data is read (in a proprietary format) into the application, the application portrays the object to the user and updates the object for the user, and finally the object is saved. This paradigm does not easily admit cooperation either with other users or with autonomous agents.

The emergence of various forms of object linking, the embedding of objects produced by one application with those of another, may change these paradigms. However, this is changing the way one looks at documents; they no longer "belong" to a single application. At present, the outcome of editing an embedded object is, in effect, to flip to that application, focusing on the embedded object, but this accommodation of the new paradigm within the old is rather strained. A new paradigm is needed where the applications, together with the users, look at and act upon the document. Such a medium-orientated paradigm will make it easier to include other users and agents in this cooperation.

3.9 Conclusions

We have seen how the AMO model has allowed us to analyse several areas, often by structuring analogies with physical examples. In particular, it has suggested an approach to adaptive interface design, and gives us a basis for evaluation of conferencing systems. However, the general approach is wider in scope than these examples. As computer systems begin to incorporate various active agents, and as we expect to work closely with other people, the electronic environments in which we work must change.

Such environments must be places where cooperating agents can work together and yet must retain the advantages of the OM paradigm. To do this, we must design the interface as a medium.

Acknowledgements Alan Dix is funded by SERC Advanced Fellowship B/89/ITA1220, and Jonathan Hassell by a CASE studentship from SERC supported by DEC.