Challenges and Perspectives for Cooperative Work on the Web

Alan Dix

at time of publication: School of Computing, Staffordshire University

http://alandix.com/academic/papers/CSCW+WWW/

Abstract

This paper investigates some of the issues which will determine the viability of the world-wide web as an infrastructure for cooperative work. In fact, taking a weak definition of collaboration, the web is already a very successful collaborative environment. In addition, it is already being used as the basis for experimental and commercial groupware. The paper takes this as a starting point and uses analytic methods developed in the field of computer-supported cooperative work to investigate the reasons for the web's present success, its strengths and weaknesses as a platform for CSCW, and prospects for future development.

Keywords: computer-supported cooperative work, world-wide web, client–server, cost–benefit, CSCW framework, user-interface architectures

1. Introduction

The world-wide web already accounts for more Internet network traffic than any other application, including email and simple file transfer. It is also a collaborative technology in a weak sense of the word – it allows people to share information. It is socially unique in that unlike the telephone system it is a broadcast medium, and unlike television and radio the users have (a large element of) control over what is published and what they see.

Most groupware systems are developed for particular platforms and are only usable within the particular organisations that use them. In contrast, the web offers a globally accessible, platform independent infrastructure. Not surprisingly, many people are looking towards the web as a potential platform for richer cooperative work, especially as hypertext infrastructure has already been the basis of several groupware systems [1]. However, it is not clear whether the protocols, servers and browsers developed for the current use of the web will be suitable for this wider use.

In February 1996 an ERCIM workshop on CSCW and the Web was called precisely to investigate these issues and brought together many people who had developed collaborative applications based on the web. The ideas in this paper were developed partly to feed into that process and partly as an analysis of the general issues which arose out of the discussion there.

The dramatic and continuing success of the web and the existing web-based groupware are the practical start points for this paper. However, these are analysed using theoretical techniques developed out of the general study of CSCW. In particular, it will make use of Grudin's analysis of the complex cost–benefit trade-off within cooperative systems, and my own CSCW framework which has been used previously as a basis for analysis and design.

I hope that this dual practical and theoretical basis will make the results of this paper both robust and generalisable. It is particularly important that any changes to the underlying web protocols and standards reflect general requirements rather than those of a particular application. The simple Shaker-like[1] design of HTTP and HTML is rapidly acquiring varied accretions and is in danger of becoming increasingly Byzantine. Recommendations of changes for improving CSCW applications must be for underlying services which make the desired high-level applications possible.

Development of this paper

The work in this paper was developed directly as a result of the ERCIM workshop on CSCW and the Web. I felt it important that the content should not be just my own ideas, but also respond to the full range of written and spoken contributions at the workshop. However, it is equally important that this paper is not simply a summary or report on the issues raised during the workshop, but represents a principled analysis of the issues, informed by the practical experience of many.

The writing of the paper has thus followed a five stage process: ① initial analysis before reading any of the materials for the workshop, using existing analytic methods and experience in CSCW, ② reading the contributions prior to the workshop with initial formulation of key issues, ③ listening to the presentations and discussions during the workshop itself, ④ presentation of the above analysis as a start point for discussion at the closing session, ⑤ final reflection and writing (now!). The mapping between these stages and the content of the paper is not simple, but the bulk of sections 2 and 4 are the result of the initial reflection ①, whereas sections 3 and 5 are due to the analysis of other workshop contributions ② & ③. As a strong proponent of theoretical analysis I was heartened to see how many of the issues that arose during the workshop had already been highlighted by the analytic approach.

Structure of the rest of the paper

The paper will begin by looking at the reasons for the web's success when viewed as collaborative technology. This will principally use Grudin's cost–benefit approach to see how the various factors have affected the critical mass point.

Section 3 looks at the way in which different collaborative systems have used the web's existing features, and have manipulated them to achieve effects not foreseen by the web's designers, or even bypassed them entirely.

Whereas these initial two sections start with the existing technology and work from there. In section 4 we will start with a general framework for understanding the role of CSCW systems which has previously been used for designing and analysing groupware under other constraints (including cooperative work over mobile telecommunications).

[1] Shaker furniture has an apparent simplicity of construction which belies the ingenuity with which form follows function in its design, which in turn reflects the simplicity and single mindedness of their religious life.

We will be able to look at each element in that model and see how well the web supports that particular feature of cooperative work.

The penultimate section will draw up a list of issues which need to be considered by those who are designing CSCW applications for the web.

Finally I'll look from one last, slightly askance, perspective at the nature of the web.

2. WWW – (some) reasons for success

Before we start adding extensions to the web, we ought to step back and look at why the web has been so successful. If we don't, then at best the features we add may not be widely used, and at worst we may undermine the web itself.

We are interested in the potential success of the web as a platform for collaborative work. In fact, the web is already a collaborative platform – we obtain information posted by others, we produce material intended for others. We'll examine it with this perspective in mind.

Critical mass

Grudin cites various reasons for the failure of CSCW systems [2]. One of the principal problems is obtaining a *critical mass* of users. Consider the cost–benefit trade-off for a user of a CSCW system. The costs of use are often constant irrespective of the number of other users. In contrast, the benefit rises with the number of other users. If you are the only user, then you don't expect much benefit from a CSCW application! So, if there is a small number of users the cost for each user is likely to exceed the benefit; only when there are a large number of other users does the benefit exceed the cost. The cross-over point is called the critical mass (see figure 1). Below the critical mass of users, the cost exceeds the benefit and so any sensible user will abandon the system, further reducing the number of users. Above the critical mass, benefit exceeds cost and so users will stay with it and others join. The challenge is getting to that critical mass position.

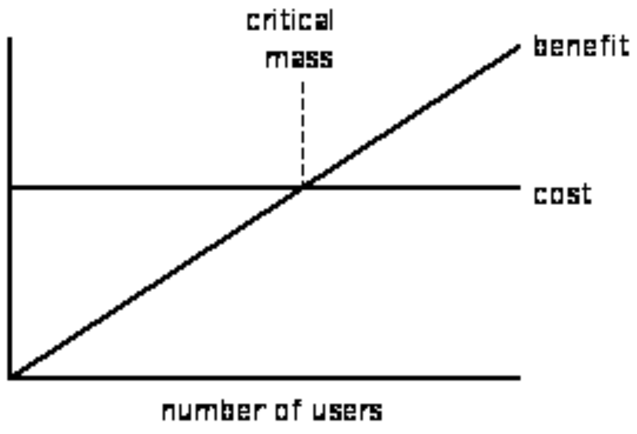


Figure 1. cost/benefit of collaborative systems

In fact, other technologies have managed to break this barrier. Most significantly the telephone which, as a mass communication tool, has many similarities with the web. As we look at the reasons for the web's success we will see that it has effectively shifted the critical-mass point in ways reminiscent of those which led to the telephone's success.

Reasons for success

Here are some particular features of the web which I believe have been critical for its success.

① *core initial user group(s)*
Initially the web was designed for a (globally speaking) small group: physicists interested in CERN's data. This was soon followed by a larger group: computer scientists who already had experience of and access to the Internet.

② *integration of existing information*
From the beginning the web was designed as a unified means of delivering existing information. Initially, this was the vast amount of on-line data coming from CERN experiments, but the multi-protocol design of the web meant that it was also possible to access the large corpus of ftp-able data. Within a short time the web also encompassed existing USENET news and information services such as gopher. Without a single line of HTML being written the web was a central access point to a vast range of on-line material.

③ *use of existing de facto standards*
Wherever possible the web has made use of existing *de facto* (rather than committee) standards. This has meant that public domain tools for construction of web materials have been readily available – especially important for image file creation.

④ *new standards – public and simple*
Where new standards have been necessary, in particular HTML itself and the HTTP protocol, they have been publicly available and designed to be as simple as possible. Even these new parts have made heavy use of existing standards; for example, HTML is a particular SGML DTD and HTTP uses content encodings originally designed for email.

⑤ *software platform: public domain, cross-platform, extensible*
The open, public standards ethos of the web has extended to many of the software products for web browsing and web page production. This was especially true during its early development, but even now (for sound marketing reasons) commercial vendors are still distributing a significant amount of free software. Even more important, the web software is neither vendor nor platform specific: web browsers are available on everything from high-end workstations to humble home micros and even over dial-up teletype lines. The information provider does not have to supply data in dozens of different formats. The importance of this has not escaped those developing groupware applications, for example, Kirby and Rodden selected the web as the infra-structure for Contact, a cooperative writing support tool because it allows them to "promote ubiquitous and heterogeneous access to the system" [3].

⑥ *cross-organisational*
One of the implications of this open, platform-independent nature of the web is that it functions as a collaborative base across organisational barriers. In contrast, most dedicated groupware products, such as Lotus Notes, are limited to the organisation that implements them, and some products may not even work across all the software/hardware platforms within a single organisation. In most Universities and

many companies the web and other Internet services are the only applications which work on all major platforms.

Reducing the critical mass

If we look at the above list, ①–⑤ all directly affect the cost–benefit trade-off.

The establishment of small cliques of users has been a key factor in the successful introduction of many groupware systems. Consider the example of telephone usage. Why should anyone buy the first telephone? Consider the situation some 30 to 40 years ago with perhaps only 1% of the general public having telephones. If this were a random 1% of the public, then only 1% of your personal friends and contacts would have a phone. There is thus little advantage to purchasing one yourself. However, the 1% was far from random. In fact, (over-simplifying somewhat!) it was the richest 1% who had telephones. If you were rich then a large proportion of your (equally rich) friends would possess a phone and hence owning one yourself was useful. The effect of this is to increase the slope of the benefit curve. If the number of users is small, but all of them are your acquaintances, then the benefit is still high. The slope is steeper and hence the critical mass much smaller. The existence of such a clique means that the technology survives and the clique forms a nucleus for growth (less rich people buy phones to talk to their richer friends ...). The physicists and computer geeks (point ①) formed just such a clique for the web and have become the nucleus for far wider growth.

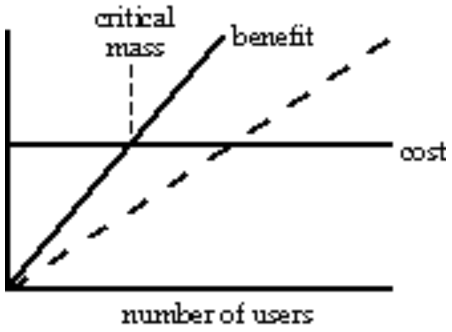


Figure 2. cliques reduce the critical mass

Point ③ has also worked by increasing the benefit of the web. However, rather than increasing the slope of the benefit–users curve, it has increased the zero point. Because the web acts as a central interface and access point to wider services, it is advantageous to use the web <u>even if no one else does</u>! Raising the zero point brings down the critical mass, possibly even to the point that one user forms a critical mass.

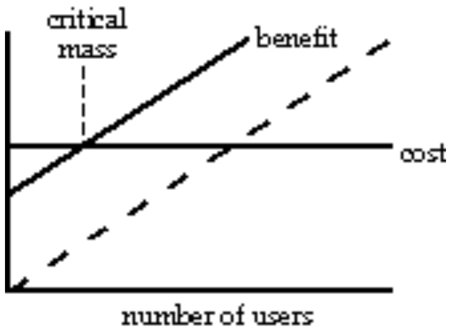


Figure 3. increasing the zero-point benefit reduces the critical mass

Finally, points ③–⑤ all reduce the cost of producing materials or viewing them. Reducing the cost in turn reduces the critical mass. Obviously low costs increase the attractiveness of any technology, not just CSCW. In fact, points ③–⑤ (especially ④ and ⑤) have been critical factors in the growth of both UNIX and X windows.

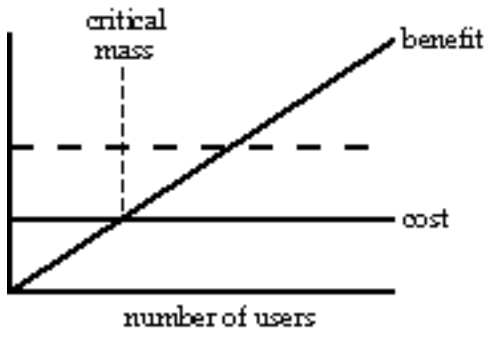


Figure 4. reducing the cost reduces the critical mass

Whose benefit?

In the above analysis I have made no distinction between information providers and consumers. In fact a problem which arises repeatedly in CSCW systems is that of disproportionate effort – those that do the work may not be the ones who get the benefit. An example, again drawing on Grudin's work, is that of shared electronic calendars [2]. Senior managers get the benefit of being able to easily arrange meetings, and everyone else has the cost of entering all their appointments into the (not necessarily easy to use) calendar system. The managers of course do not have to actually <u>use</u> the system themselves as their secretaries do the work!

If we distinguish between providers and consumers on the web we may get different cost–benefit balances. In the above example, employees may be forced to use a system for which they perceive little advantage. In the case of the web, use is usually voluntary and if either providers or consumers do not perceive any benefit they will simply not use it.

For consumers the benefit is obvious: they get information they require. The costs are in obtaining an Internet connection (apparently free for much of the initial academic user group) and in the time spent (or wasted) surfing and searching.

The costs for the provider are in the production of pages themselves and the maintenance of or renting of space on a web server. The benefits are perhaps less clear. Indeed, the director of a privately run historical exhibition told me that although they were paying a bureau to put some pages on the web he didn't see any benefit of anything beyond a minimal presence. Was he right? Normally in the marketplace providers of products or services get their benefit from the monies paid. In the early days of the Internet, the main providers were academics who perhaps regarded the free dissemination of information as a sufficient end in itself, especially as part of a global quid-pro-quo. For commercial providers (and to be honest a lot of academics) this is not a sufficient end. Instead their main benefit has been exposure – a relatively cheap form of marketing. For many this has not been sufficient hence the increase in the use of electronic transactions, to close sales while the marketing message is still strong and subscription services. We can sum up this progression of benefits for the provider as philanthropy, publicity and direct profit.

Whatever the overall merits or demerits of the web, in a global free market its ultimate success or failure depends on appropriate aggregate benefits for each individual. As the nature of the consumers and providers on the web changes the nature of that balance has to keep pace. Of course for CSCW systems on the web the provider/consumer distinction is not very meaningful, instead we have collaborators with (possibly) shared goals. As seen in the example above, this does not remove the issue of disproportionate effort, but does change its nature. The growth of web-based CSCW system could form part of the re-alignment of the web in an increasingly commercial environment.

Good or not?

Whether all the factors identified above are good or not may be a matter of opinion. However, the fact that the web can act as a platform-independent, cross-organisational CSCW infrastructure is why the ERCIM workshop was run and why this special issue exists! Amongst other things, the cross-organisational nature means that an organisation can take advantage of the wide variety of information available, and publicise itself using the web, even before a critical mass has formed within the organisation. However, whether encouraging collaborative relationships to develop across organisational boundaries is regarded as good, will depend very much on the culture and philosophy of the organisation. In some sectors this cross-organisational interchange was already well established using EDI and similar technologies [4], on the other hand in this special issue, Ginsburg and report that one company they studied "place little or no value in interoperating with external entities" [5].

Also, the UNIX–X pattern may not be the only model, nor even the best. Contrast this with the DOS model where a single powerful vendor (IBM) drove the market place by sheer force of presence. Possibly the same situation may arise with the web as, for example, Netscape and Microsoft establish vendor-specific web extensions. Vendor-specific solutions may be favoured in commercial environments: even if the initial financial costs are higher, this will be offset by greater perceived benefits in terms of support services and correspondingly lower staff costs. This is perhaps one of the reasons why Windows NT is gaining ground against UNIX as the preferred server operating system.

Arguably, the time for a single dominant player has been missed (for good or ill), and the web is solidly on the UNIX path. However, remember that no matter how successful UNIX has been, the path has often been rocky. The very features of its success – publicly available source code allowing multi-platform development – also led to a period of fragmentation, with multiple incompatible variants of UNIX springing up hydra-like throughout academia and industry. This has been followed by a period of standardisation and a move towards 'controlled openness', a phase which is still incomplete. The current proliferation of different HTML variants (including various extensions proposed in the ERCIM workshop) seems to be following this pattern. However, whereas many UNIX users spend most of their working time with one variant, the very strength of the web is that access is global. The web may not be able to withstand such fragmentation. Standing against this danger is the wide commercial recognition of the need for open standards. Whether fragmentation or standardisation will be stronger in the medium term remains to be seen.

3. Using the Web for CSCW?

The web was designed principally as a mechanism for information access – to some extent a form of collaboration in its own right. However, it is not at all obvious how this existing infrastructure is best used for richer forms of collaborative activity. Several solutions to this are possible and are evident in the systems presented at the workshop and described within this report.

The most obvious architectural issue is which parts of the web infrastructure are modified or extended. This infrastructure operates in three parts: server, client and protocol:

- *server-end modifications*
  Several systems including BSCW [6] and futplex [7] use server-end extensions: both CGI scripts and independently running servers. The former make use of the existing ease of extension within web servers.
- *client helpers and applets*
  The incorporation of JAVA Netscape and other web browsers has emphasised the value of client-end computing, especially for rapid user interface feedback. In common with server-end scripting, the need for users to download special code is avoided. This will clearly form a central part of many CSCW systems on the web [8]. In addition, some systems have made use of downloaded helper applications and modified clients, including clients modified to run Tcl/Tk as a client-side script language [9].
- *additional protocols*
  HTTP is certainly not suited to real-time conversations! Hence, those systems which aim to give 'talk' style interaction must use their own protocol over independent Internet channels [8, 9]. The same need has arisen in the use of real-time audio and video [10]. The question here is which of these extensions should be incorporated into future versions of HTTP and which should remain as separate protocols. For real-time audio and video the choice is clear, as they have completely different transport layer requirements to each other and to the web (in fact the issue is whether the TCP/IP suite needs extension). As one first considers interactive text messages, and then notification mechanisms, the issue becomes far less clear.
- *HTTP as a transport layer*
  The systems which fall into one or other of the above categories are all clearly 'web' applications. In contrast, Alliance [11] uses neither HTML, nor (modified) web clients. Instead, it has its own data formats (a form of SGML) and user interface (a structured SGML editor). It uses the HTTP protocols to communicate with CGI scripted servers, but the results are handled in a completely non-web fashion. It effectively uses HTTP as a transport layer that allows very large (whole file) transactions.

As well as looking at what parts of the web infrastructure are used by an application, we can ask what the web is being used for.

- *interface to an application*
  Many systems use the web as an interface. It is hard to produce platform-independent interfaces. The web combines the advantages of platform independence with a rapid distribution mechanism. Furthermore, interface

development is iterative, and using the web means that changes are automatically propagated to all users. The meaning of 'user interface' in relation to the web is quite interesting. For example, in the BSCW system architecture, the word 'interface' is on a box sitting on the server side of the Internet. This is a sound use of the term 'interface' separating the surface interaction (including presentation, scrolling and text editing), from the deeper levels of interaction [12]. Interface toolkits can be too constraining and this is where client-end code is being used to generate special interface effects.

- *rapid prototyping engine*
  Because the web handles much of the surface interaction it can be used as a prototyping engine in a similar way to the use of HyperCard or Visual Basic. Even if the final product may be engineered in a different programming and interface environment, web prototypes can allow easy demonstrations and proof of concept. More important from a CSCW point of view, the ease of extension on both server side and client side makes the web an excellent groupware prototyping platform. As Alliance demonstrates, this is true even viewing the web merely as a transport layer!
- *just another word for the Internet?*
  Several systems bill themselves as being 'web' based, but in fact establish independent Internet connections to specialised servers or other clients. In some cases this is to augment the pure web part of the application (for example, supplying notification services). In others the application, once launched, is entirely separate from the web. The 'talk' applications are examples of this. Are these truly 'web' applications, or are we in danger of following the popular press and confusing 'Internet' with the 'web'?
- *access point*
  In fact, for the talk-style applications, although the web is not required <u>during</u> a chat, it is essential for <u>establishing</u> communication. There is no need to know the Internet address of the remote user's machine or a chat server; instead one can simply navigate to an appropriate page on the web, then click and connect. Video and audio links fall into a similar category. When combined with web pages that keep track of the location of a user and reflect the user's current status, this can revolutionise personal communications, making phone tag a thing of the past! In general, instead of seeing extensions to the web, we perhaps ought to be thinking of the web as a central access point to information, applications and services. At a global level this would reflect the move towards document-centred personal computing as seen in Apple's OpenDoc™ architecture.

4. CSCW in context

In the last two sections we have looked first at the web itself, and then at existing CSCW applications on the web. In this section we will start with the nature of CSCW itself and ask to what extent the web could support it. We will use a framework developed some years ago as a way of exposing general groupware issues [13]. It has been used since as a tool for design and to analyse other CSCW issues. In particular, it has recently been used to examine the suitability of mobile telecommunications as a basis for CSCW technology [14]. This has some similarity to web-based collaborative applications, in particular the need for local processing for rapid feedback.

The framework is based on examining the interactions between the participants in cooperative work and the artefacts upon which they work. The elements of the framework will be introduced incrementally through this section.

Communication and control

In order to talk about cooperation we must have at least two people cooperating. These participants are represented by the two circles labelled 'P' in figure 5. In order to coordinate their activities they will typically communicate directly in some way. In addition, if they are engaged in cooperative work, then there will typically be things on which they are working, either physical or electronic (as is most common in computer applications). These are labelled 'A' in figure 5 – the artefacts of work.

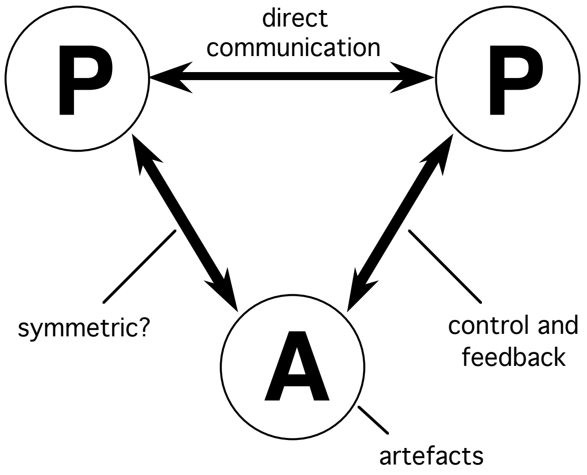


Figure 5. communication and control

The importance of shared artefacts, often simply pieces of paper, as the focus of coordination is of increasing importance in CSCW literature, recognised especially in ethnographic studies [15, 16]. For example, small pieces of card called flight strips are used in air-traffic control to record aircraft position and other details. However, from their studies of these Hughes et al. say that they are "more than just a repository of information" but serve many purposes both for individuals and the team being "a resource with which to make sense of what is going on" [17]. The presence of paper and other items in the work environment has also been found to be the key `triggers' for prompting activities as part of long-time scale collaboration [18].

In the web the artefacts are usually the web pages or electronic documents which we are handling. Direct communication is not catered for at all. For this reason, several CSCW applications supply direct communication facilities, either synchronous or asynchronous. Examples of the former include HushTalk [9], supplying talk-style facilities, and 3D Dive's shared virtual reality [19]. Asynchronous communication is primarily supplied using a bulletin board style transforming communication into an information structure, which the web is designed to manage. Thus the examples of asynchronous communication require less 'breaking' of the web model than the synchronous ones, which effectively bypass the web protocols entirely. This again raises the question of whether we should look for extensions to the web protocol to better manage synchronous applications, or whether we should always expect these to use different protocols.

When we look at the artefacts themselves, the natural domain of the web, we see a strong asymmetry. In some working circumstances all parties have equal access to the shared artefacts upon which they are working. Often only one person works on them at any time, but that individual may vary. Because of the distributed nature of the web, the access is far less symmetric. The web page is normally stored on a computer 'close' to the author. The author will typically edit the file using interactive tools, possibly involving some uploading/downloading of files from the web-server to a local machine. In contrast, the reader of the page will only be able to read it but not update it in any way. Various forms of shared document repository such as Dress [20], futplex [7] and BSCW [6] are ways of redressing this asymmetry. In addition, they often make the job of authoring information easier, as the upload/download cycle can be quite complex on some systems.

Meta-information: understanding and deixis

In order to be able to communicate we need at least some level of common understanding, at very least a common language and beyond that some degree of common culture and knowledge. In addition, when using technology to communicate, we need to understand the nature of the medium and the way in which it shapes our conversations. By providing a common interface the web gives some unity to collaborative work. However, the unity of presentation is often far less than users imagine. In other CSCW applications problems occur when participants see similar, but different, information on their screens [21, 22]. In addition to this technology-based difference in perception, the very global nature of the web means that one is likely to communicate with people with whom one shares little common background and perhaps little common expectation of behaviour. For example, in the context of close working colleagues issues of security and access control may be treated lightly, whereas your trust in the entire global Internet community may be less strong!

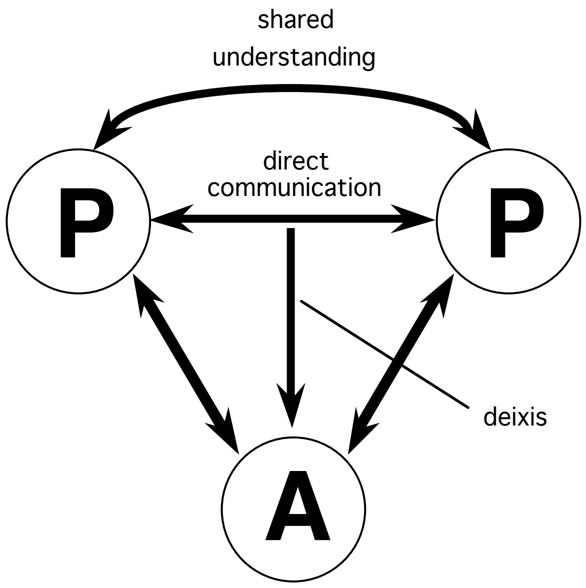


Figure 6. meta-information

In conversation we continually refer to things either by name, description, pronoun, or pointing. This is called deixis or deictic reference. A frequent problem in groupware is that it is often difficult or impossible to reference the work domain from the conversation domain. For references to entire documents, the URL offers an ideal method of embedding such references and this is used by many systems, especially where the messages may themselves be treated as HTML source. URLs are still not ideal for references to locations within documents and are not as stable as one would like.

Feedback and feedthrough

You interact with an object to control its behaviour and update its state. In an interactive setting one now normally expects rapid feedback on one's actions – I am typing and the letters appear in the document at once. In order to make feedback as fast as possible one normally tries to bring the data as close as possible to the person doing the editing. In the case of web pages this usually means uploading HTML source to edit locally.

In a cooperative setting not only is it important to see one's own updates, but also to see the effects of other people's actions. This is feedthrough. The presence of feedthrough effectively creates an additional channel of communication through the artefacts themselves. In real life, cooperating over physical objects, this communication through the artefact is often more important than direct communication. For example, imagine you are moving a large piano. You may say things to each other – "move your end up a bit", "careful of the step" – but in fact the most important thing is the feel of the other person's movements through the movements of the piano. This sort of communication is effective partly because it is tied so intimately to the work itself, and partly because it is implicit, unconsciously noticed and acted upon.

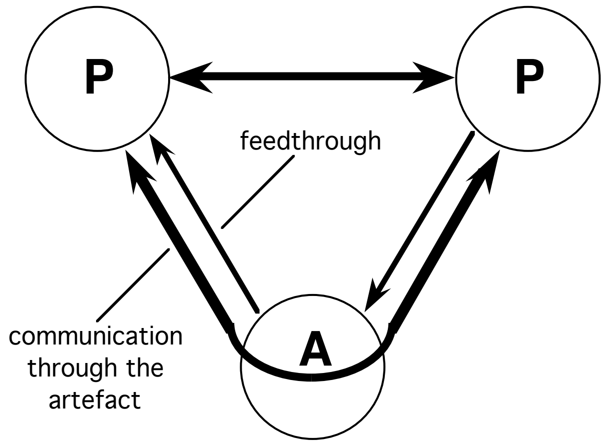


Figure 7. feedback and feedthrough

Feedthrough is often weak in electronic cooperation, and this is worsened by the need to make objects 'close' to the person updating them. This is certainly the case in the web, as we saw earlier when asymmetry of access was discussed. However, effective feedthrough is essential to fluid collaboration and must be a major issue for any cooperative application.

Awareness

The word awareness is used frequently within CSCW [23]. Sometimes it refers to awareness of the presence of other people. That is, awareness of who is around and their availability for cooperative activity (see fig. 8). Feedthrough is also a form of awareness, in this case awareness of what has happened. However, there may often be several possible causes of a change and in order to complete the picture we need awareness of how the change happened, which, together with our conversation with other people and understanding of the context, allows us to infer why it happened.

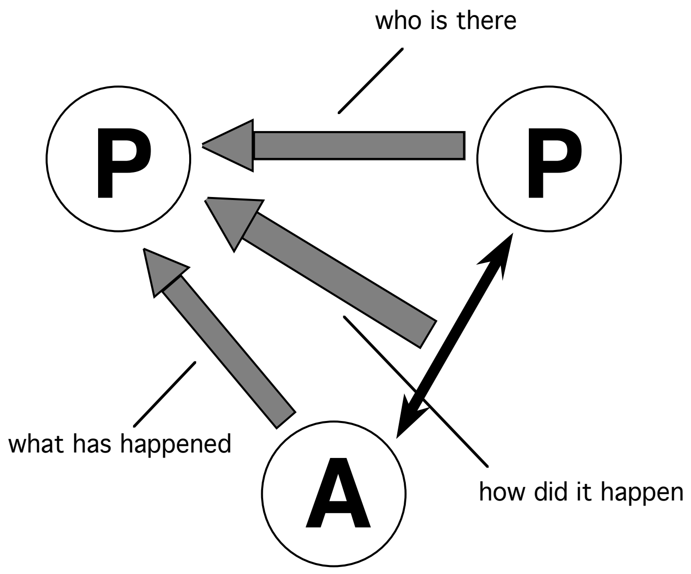


Figure 8. awareness

Two interesting developments in this area has been the development of formal and semi-formal model of awareness in CSCW [24, 25], and also the development of infrastructure support for awareness in non-web based groupware .[26]

The web is already an important way in which people get to meet each other, often by browsing one another's home pages. In fact, while I was writing this paper I was contacted by another 'Alan Dix' in the US who had found my home page. This is augmented in systems like @Work [10] with extra information about availability and methods of contact. Various kinds of non-web based virtual spaces, often implemented using video or constant audio connections, have been developed over many years and been the subject of extensive study [27-29]. More recently this has been matched by an explosion of virtual meeting places on the web, many simply recreational, others constructed as part of research programs such as the Internet Foyer at Nottingham [19] and those built upon the CommunityPlace infrastructure described in this special issue [30] (which is based directly on the semi-formal models of awareness mentioned above). Caching can make such virtual locations difficult to implement as 'visitors' served from caches will not be seen by the site itself.

'How it happened' awareness is perhaps the most difficult to manage over the web. It is all about the occurrence of actions in time, implicitly noted and understood while they happen, but so hard to reconstruct afterwards. Even when you perform actions yourself it is very difficult to recall the precise reasons for each step and the order in which they occurred. Systems providing shared workspaces typically record who has made updates and when. This supplies some of the 'how it happened' information. However, the observer must try to reconstruct the situations from an explicit trace. This may be made easier if the person who performs an update is forced to add some explanation, as is often the case in version management systems. However, both the person performing the update and the observer are forced to represent explicitly things which are normally implicit. In fact, the very thing which differentiates 'awareness' from 'observation' is that it is implicit and low effort – very difficult to obtain for temporal information in an asynchronous environment. Awareness of `how' and 'why' problem fin all asynchronous groupware. One solution is to record the activities of individuals or group interaction and later replay the log to those who were not `present' (in the virtual sense!) [31]. In a web environment this is made more complex as the transactional nature means that servers would only be able to record activity at a very course level without assistance from client software.

The different forms of awareness have several common issues. First of all the pace of feedthrough or awareness is very important. Elsewhere I have argued strongly that pace (how often one interacts) is usually far more important than bandwidth (how much one communicates) in collaborative systems [32]. In the case of feedthrough, how long is it from when a change takes place until it is observed by someone else? The answer to this is partly dependent on the technology (e.g., the latency of the Internet, how often one machine polls another) and partly on the behaviour of the participants (e.g., how often changes are uploaded, how often the other person visits a page).

A second crucial issue is that of initiative. Who or what makes awareness possible? For normal web pages, initiative lies very much with the observer. A change will (or may) be noticed only when the page is next visited. In a very large information space this may never happen! Web-based repositories and bulletin boards get round this by marking high-level aggregates (conferences or folders) when any item within has been changed, allowing the user to tunnel in to the changed items. The initiative is still with the observer, but is aided by the objects themselves. The pace of such a style of interaction is still too low for some activities and so some explicit notification is supported. Unfortunately this is an area in which the web is particularly weak, because it has largely stateless servers. There are lots of sound reasons for this, but it does rule out notification services. Those systems which supply user-level notification must therefore handle it entirely themselves, by running separate notification servers and then, when an update occurs, sending an event in real-time or by email to interested parties.

The flip side to initiative is interruption. If other people or things have the responsibility for telling you when things happen, they may tell you when you least want them to! Think of those hundreds of email messages telling you about insignificant changes, or dialogue boxes appearing and beeping at you when you are in the middle of composing that really difficult letter! The secret of pace is matching the pace of communication to the appropriate pace of the cooperative task. Different tasks have different paces and so the same notification mechanism will not be universally applicable (beeping dialogue boxes may be useful when the machine is about to shut down). At present the web is far too slow (infinitely slow) at notification. Extensions must be careful not to overshoot the mark.

5. Issues

We will now summarise some of the main issues arising from the rest of the paper, plus a few not covered by the structured analysis.

- structure of information
  The importance of shared information was highlighted by the CSCW framework, but not the range of options for its structure. At present the web knows of only one level of structure – whole pages! However, the systems developed for CSCW on the web incorporate additional levels: some recognise directory/folder hierarchies,

treated as conferences in bulletin boards; some have a more explicit model of the web network structure; others look inside and recognise portions of pages. In fact it is not clear what should constitute a 'document' on the web. SGML assumes that the document resides in a single file, but the delays on the web, and its Hypertext structure, suggest that one breaks up large documents into fragments. At this point it is clear that the page is by no means the entire document. Things are further complicated by CGI scripts and special servers which generate pages dynamically. At one level the web purports to be a Hypertext, but many CSCW applications are treating it more like a user interface. If dynamic pages proliferate without any additional features (for example, 'methods' on CGI scripts for indexing and connectivity) then automatic web indexers and spiders will cease to function – the shared information spaces meant to foster cooperation will be opaque to the outside world.

- availability and replication
  The need for rapid feedback and constant availability points clearly towards increased use of caching and replication. However, the use of dynamic pages militates against this and web caches are likely to fail to operate efficiently on shared repositories based on CGI scripting.
- writing as well as reading
  As discussed, the normal nature of the web is to have local update and remote browsing. The desire for more symmetric remote access leads inevitably towards issues of concurrent update, locking, access control and version management. We can see these issues starting to be addressed in existing systems and they will surely become more critical. Also, many of the developers of CSCW systems have complained about the lack of upload facilities, including the fact that most servers do not implement the HTTP 'PUT' method.
- security and authentication
  As well as making things easier for remote users, we may also want to make things harder! In Ginsburg and Duliba's study one company sited poor security as the main barrier to using the Internet [5] a view shared by many outside (and even within) academic environments. Security was sited as one of the main reasons Security issues are already firmly on the web agenda. However, security and protection mechanisms may easily become fossilised in the web along similar lines to those in traditional operating systems. It is far from clear whether these are the appropriate mechanisms for cooperative systems.

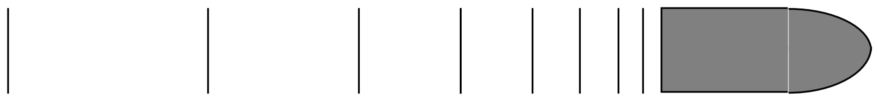
  It is interesting to note that security, although an issue which quickly springs to mind with the web, was not prompted by the analytic methods. The CSCW framework reflects the emphasis in much of CSCW towards making access and awareness easy rather than restricting it. This is part of a general orientation to ideas of openness within the CSCW community, but it is recognised that not all collaborative enterprises are necessarily totally friendly [33].
- notification and awareness
  It is clear that an effective general purpose notification mechanism is required either running alongside the web or built into the web protocols. Without such a mechanism feedthrough and awareness cannot be effectively developed except by using non-standard add-ons. One such path, already being adopted for non web-based applications, is the use of separate servers, for example, work at Lotus. on

separate notification servers [34] and GroupDesk and event distribution and awareness server [26].

- who are the users?
  The web is designed for global access. Potentially the users of a web page could run into hundreds of millions. Most CSCW systems have a smaller target user group (!): either a small group working closely together on a project or an individual organisation. As we saw in section 2, the web offers the possibility of cross-organisational collaboration – if that is wanted. Zeno is an example of this wider perspective, supporting decision making between government, commercial and community groups [35]. It is important that one is clear about the target user group as, for example, a document repository for public use will have different requirements from one installed for private use in a small company behind firewalls. Further complicating the picture are automatic agents and traders (as, for example, in AlephWeb [36]). Are these possible 'users' of a groupware system? If so, then as discussed previously we may need to be very careful about our use of caches and dynamic pages.
- user interface description
  The HTML mark-up is designed primarily for static pages – it is a <u>hypertext</u> mark-up language, not an <u>interface</u> description language. Forms add some element of user interface description and this is being augmented by JAVA and client-end processing. The mark-up must work with different browsers, window sizes, screen resolutions etc. – by no means an easy task! One of the problems cited by several web interface developers is the frustration caused by this lack of formatting control. In fact, whereas SGML mark-up was precisely defined many years ago, the language for describing the formatting of SGML documents languished. If describing the device-independent formatting of static documents is so difficult, what hope for interactive pages! Actually, this problem is not confined to web interfaces; for example X Motif has constraint-based layout widgets because X programs have to cope (or fail to cope) in a similarly hostile display environment [37]. On the other hand, it should be noted that many interface developers find using these constraints very difficult! Of course, the web is not the only interface; for example, the BSCW project is considering having web interfaces for public access and separate non-web interfaces for local access and system administration functions. This trend is also reflected in the growing number of commercial database and similar packages which have web publishing facilities, but use their existing update mechanisms. Contrariwise, web pages are increasingly being used as a user interface to configure network servers both hardware and software, precisely because it means that virtually any terminal, PC or workstation can be used.
- architectural issues
  User interface architectures, such as the Seeheim model [38] and MVC [39], can afford to be rather vague, since when interface and application reside in the same program they can both, in the end, access anything! In contrast, distributed interfaces demand a much clearer idea of what goes where and how the parts communicate. We can see examples of systems where the 'interface' sits firmly on the server side (such as BSCW) and the web browser is being used as a presentation manager. On the other hand, some systems effectively put the whole application on

the client side (as a helper or downloaded applet). Many of the applet-based systems struggle against the lack of peer–peer communication in the web. As well as affecting some of the other issues we have discussed, such as the pace of feedthrough and feedback[2], architectural design is critical in determining the scalability of a system.

- integration
  One vision we have discussed is the role of the web as a universal access mechanism, rather than a universal interface. Where the web is inadequate for CSCW applications (especially synchronous services such as audio/visual connections and on-line chat) it may be better to develop specialist protocols in which the web acts as the unifying medium, rather than further clutter the basic protocols. The web would then act as a global equivalent of the desktop metaphor, indeed the idea of 'my computer' may fade as stand-alone web-client engines act as the omega point along the diskless workstation path. An alternative is to see the web as standing alongside other services and to look for a unifying framework for them all, a sort of Meta-WWW.
- beating the speed of light
  The rate of growth of the web, both in volume and in new and changing technology, may mean that we have to run even to keep up. One researcher at the ERCIM workshop had funding for a web-based project, but was worried that if he waited six months to start the project it might be out of date before he started! For the commercial software developer/consultant this is a two-edged sword: on the one hand your clients need you to keep them up to date, on the other hand you may have trouble keeping up to date yourself! For the researcher the focus must be on the more fundamental issues raised by their work, as the practical aspects are likely to be passé by the time they are published.


faster than a speeding bullet?

6. A last word

As part of the work towards a contribution to the Advanced Visual Interfaces AVI'96 conference, I have been led to consider the nature of non-visual interfaces [40]. Aural interfaces are well established for both sighted and blind users [41]. However, watching my dog one day made me consider what the world is like if your primary sense is smell. When you look around you, you see a snapshot of all of space (fading into the distance) at one moment. However, if you want to know what happened ten minutes or one hour ago you have to remember. Now imagine you are a dog with a sensitive nose. You smell at the base of a tree and from the different scents and the way they have aged you can tell what animals have visited there and how long ago. That is, you get a snapshot of all time (fading into the past). If you want to know what has

[2] Ongoing work on the influence of web archtecture on timing issues can be found at:
`http://www.soc.staffs.ac.uk/~cmtajd/topics/webarch/`

happened at a different tree that you have recently sniffed, you have to remember. With vision you perceive all of space at one instant and use memory for time, with smell you perceive all of time at one instant and use memory for space!

Clearly modern desktop interfaces are strongly visual; in principle you always see an up-to-date view of the contents of the folders on your desktop, or the current page of the document you are editing. However, you have to remember what it was like some time ago. In contrast, an old command line interface such as UNIX shell or the DOS command interpreter gives you a trace of commands you have executed and the results – a view of your history through time; but you must either explicitly ask, or remember, what files are in a directory – a nasal interface. In fact, both can be useful and the lack of history in a graphical interface can be just as problematic as the lack of immediate context in a command line interface.

If we consider the web, it is point based – you visit individual pages within an interlinked network. One rarely gets an view of the overall state of the system: the focus is on a single location. Several CSCW systems are adding information to pages or documents specifying who has visited them, when they did so and what they did there. Rather like sniffing at trees.

7. Acknowledgements

Thanks to GMD FIT for funding my visit to the ERCIM workshop at Bonn and to all those at the workshop and subsequently whose ideas and discussions I have mined for this paper.

8. References

1. Haake, J.M. and B. Wilson. *Supporting collaborative writing of hyperdocuments in SEPIA*. in *CSCW'92*. 1992. Toronto, Canada: ACM Press. p. 138–146.
2. Grudin, J., *Why CSCW applications fail: problems in the design and evaluation of organisational interfaces,* in *CSCW'88 Proceedings of the Conference on Computer-Supported Cooperative Work*. 1988, ACM SIGCHI & SIGOIS: p. 85–89.
3. Kirby, A. and T. Rodden. *Contact: support for distributed cooperative writing*. in *ECSCW'95*. 1995. Stockholm, Sweden: Kluwer Acadamic. p. 101–118.
4. Schill, A., *Cooperative Office Systems*. 1995, Prentice Hall.
5. Ginsburg, M. and K. Duliba, *Enterprise-level groupware choices: evaluating Lotus Notes and web-based solutions*. Journal of CSCW (special issue in CSCW & Web), 1996. .
6. Bentley, R., *et al*. *The BSCW Shared Workspace System*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.
7. Holtman, K. *The futplex system*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.
8. Walther, M., *Supporting Development of Synchronous Collaboration Tools on the Web with JAVA*. Journal of CSCW (special issue on CSCW & Web), 1996. .

9. van Welie, M. and A. Eliëns. *Chatting on the Web*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

10. Sandor, O. and K. Tollmar. *@Work, the design of a new communication tool*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

11. Decouchant, D. *Alliance: A structured Cooperative Editor on the Web*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

12. Took, R. *Surface interaction: a paradigm and model for separating application and interface*. in *CHI'90 Conference Proceedings – Empowering People*. 1990. ACM Press. p. 35–42.

13. Dix, A.J., *Computer-supported cooperative work — a framework,* in *Design Issues in CSCW,* D. Rosenburg and C. Hutchison, Editors. 1994, Springer Verlag: p. 9-26.

14. Dix, A.J., *Cooperation without (reliable) Communication: Interfaces for Mobile Applications*. Distributed Systems Engineering, 1995. **2**(3): p. 171–181.

15. Luff, P., C. Heath, and D. Greatbatch. *Tasks-in-interaction: paper and screen based documentation in collaborative activity*. in *Proceedings of CSCW'92*. 1992. Toronto, Canada: ACM Press. p. 163– 170.

16. Rouncefield, M., *et al*. *Working with "Constant Interruption" CSCW and the Small Office*. in *Proceedings of CSCW'94*. 1994. Chapel Hill, North Carolina: ACM Press. p. 275–286.

17. Hughes, J.A., D. Randall, and D. Shapiro. *Faltering from ethnography to design*. in *CSCW'92*. 1992. Toronto, Canada: ACM Press. p. 115–122.

18. Dix, A.J., D. Ramduny, and J. Wilkinson. *Long-Term Interaction: Learning the 4 Rs*. in *CHI'96 Conference Companion*. 1996. Vancouver: ACM Press. p. 169–170.

19. Brown, C. and S. Benford. *Collaborative Visualization of Large Scale Hypermedia Databases*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

20. de Bra, P. *Dress: a simple Document Repository Service Station*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

21. Stefik, M., *et al*., *Beyond the chalkboard: computer support for collaboration and problem solving in meetings*. CACM, 1987. **30**(1): p. 32–47.

22. Stefik, M., *et al*., *WYSIWIS revisited: early experiences with multiuser interfaces*. ACM Transactions on Office Systems, 1987. **5**(2): p. 147–167.

23. Dourish, P. and V. Bellotti. *Awareness and coordination in shared workspaces*. in *CSCW'92*. 1992. Toronto, Canada: ACM Press. p. 107–113.

24. Rodden, T. *Populating the application: a model of awareness for cooperative applications*. in *Proceedings of CSCW'96*. 1996. .

25. Benford, S., *et al*. *Managing mutual awareness in collaborative virtual environments*. in *Proceedings of ACM SIGCHI conference on Virtual Reality and Technology – VRST'94*. 1994. Singapore: ACM Press.

26. Fuchs, L., U. Pankoke-Babatz, and W. Prinz. *Supporting cooperative awareness with local event mechanisms: the GroupDesk system*. in *ECSCW'95*. 1995. Stockholm, Sweden: Kluwer Acadamic. p. 247–262.

27. Heath, C., P. Luff, and A. Sellen. *Reconsidering the virtual workplace: flexible support for collaborative activity*. in *ECSCW'95*. 1995. Stockholm, Sweden: Kluwer Acadamic. p. 83–99.

28. Takemura, H. and F. Kishino. *Cooperative work environment using virtual workspace*. in *Proceedings of CSCW'92*. 1992. Toronto, Canada: ACM Press. p. 226–232.

29. Olson, M.H. and S.A. Bly, *The Portland Experience: a report on a distributed research group,* in *Computer-supported Cooperative Work and Groupware,* S. Greenberg, Editor. 1991, Academic Press: p. 81-98.

30. Lea, R., Y. Honda, and K. Matsuda, *Virtual society: collaboration in 3D spces on the Internet*. Journal of CSCW (special issue on CSCW & WWW), 1996. .

31. Manohar, N.R. and A. Prakash. *The session capture and replay paradigm for asynchronous collaboration*. in *ECSCW'95*. 1995. Stockholm, Sweden: Kluwer Acadamic. p. 149–164.

32. Dix, A.J. *Pace and interaction*. in *Proceedings of HCI'92: People and Computers VII*. 1992. Cambridge University Press. p. 193-207.

33. Easterbrook, S., ed. *CSCW: Cooperatlon or Conflict?* 1993, Springer-Verlag: .

34. Patterson, J.F., M. Day, and J. Kucan. *Notification servers for synchronous groupware*. in *Proceedings of CSCW'96*. 1996. .

35. Gordon, T. *Zeno: a Mediation System for the World Wide Web*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

36. i Mulà, G.R. and L.N. Moldes. *AlephWeb: a CSCW Large Scale Trader*. in *ERCIM workshop on CSCW and the Web*. 1996. Sankt Augustin, Germany: GMD/FIT.

37. OSF, *OSF/Motif Programmer's Guide, Revision 2*. 1995, Open Software Foundation, Prentice Hall.

38. Pfaff, G.E., *User Interface Management Systems*. 1985, Springer Verlag.

39. Myers, B.A. and D.S. Kosbie. *Reusable hierarchical command objects*. in *CHI 96*. 1996. Vancouver, BC, Canada: . p. 260–267.

40. Dix, A.J. *Closing the Loop: modelling action, perception and information*. in *AVI'96 – Advanced Visual Interfaces*. 1996. Gubbio, Italy: .

41. Edwards, A.D.N., ed. *Extra-ordinary Human–Computer Interaction*. 1993, Cambridge University Press: Cambridge.