# Cooperation without Communication: the problems of highly distributed working

Alan Dix

**hci@hud** – The HCI Research Centre
School of Computing and Mathematics
University of Huddersfield
Queensgate
Huddersfield, HD1 3DH
A.J.Dix@hud.ac.uk

**Versions of this paper and subsequent work**

This paper was first distributed in 1992 and has since been cited in several places, but has never been formally published. Some of the material has found its way into the CSCW sections of the textbook "Human–Computer Interaction" (Dix et al. 1993) and into related papers under development (Dix and Beale 1995; Dix 1995a). However, the foci of these publications are different and so some of the material in this paper does not appear elsewhere. The paper is therefore now being published in report form in order to make it more easily accessible. As it has been cited in its original form I have not attempted to make major alterations, except to make typographical corrections and update a few references. The general thrust of the argument still stands, but there are some omissions when considered in hindsight. Most notable is the lack of any reference to the CODA distributed filestore project at CMU (Satyanarayanan et al. 1990). The project had been underway for some years when the paper was written, but I did not become aware of it until later. Also the paper does not discuss Lotus Notes. This was available then but did not at that time have the market and academic exposure it does now.

# Cooperation without Communication: the problems of highly distributed working

Alan Dix

**hci@hud** – The HCI Research Centre
School of Computing and Mathematics
University of Huddersfield
Queensgate
Huddersfield, HD1 3DH
A.J.Dix@hud.ac.uk

**Abstract**

The aim of this paper is to highlight the problems of *highly distributed workers*: mobile workers, tele-workers and those separated by slow networks. These are characterised by the intermittent nature of their communications. The standard time/space matrix is extended in order to adequately describe these workers, in the process revealing the ambiguity of the term 'asynchronous' as applied to groupware. Two specific problems areas are discussed: establishing context in direct communication and re-synchronisation of different versions of shared objects after periods of concurrent working. Nevertheless, patterns of individual work in these settings suggest a strong potential market for groupware.

**Keywords**: mobile computing, tele-commuting, version control, asynchronous work

## 1. Introduction

This paper aims to raise awareness of the importance of a class of workers, principally mobile and tele-workers, engaged in what I call highly distributed work, or simply *distributed work*. The term distributed in CSCW literature sometimes means simply geographically distant, even when communication is strong and thus sites are logically local. In this paper distributed takes a stronger meaning, namely where communications are infrequent and thus where workers are concurrently engaged in common tasks whilest unable to communicate directly. Distributed workers have problems of communication and coordinating shared work, but offer opportunities for the design of effective groupware.

Some related papers describe the problems of distributed working from the point of view of the single-user (Dix and Beale 1995), discuss the specific issues for mobile computing using wireless networks(Dix 1995a) and propose the use of a form of version control as an infra-structure to support highly distributed co-authoring (Dix and Miles 1992).

## 2. The workers

### 2.1. Mobile workers

The image of computers during the 60s and 70s was probably the whirling tape drive, during the 80s the desktop computer. For the 90s the corresponding image is likely to be the notebook computer. A train-ride hardly seems complete without some executive working with computer and portable phone (to the muted strains of Verdi from a Walkman). Portable computers are projected to surpass the sales of the desktop computer (and perhaps by now they have). This growth of the market has been effectively technology limited as it is only in the last few years that sufficiently compact and low power components have become available, but has clearly developed a momentum all of its own. The software for such computers is either identical to that for its deskbound cousin (standard word-processors, spreadsheets etc.) or 'grown up' versions of that available for pocket organisers (alarm clock, appointment diary, address book). There is but little support for integration with other systems: principally file transfer programs and (often hard to use) communications packages.

### 2.2. Home-workers and tele-workers

Computers are not only found on the train, but in the home. Bringing work home used to refer to a briefcase of papers, it increasingly means a floppy disk or two. For such overnight work the home computer looks very like a mobile computer, file transfer is via floppy disk rather than serial line, but otherwise, logically very similar. (I often use a program intended for file transfer between computers to copy data to and from floppy disk).

As the period at home becomes larger, the worker becomes a tele-worker or tele-commuter. Most such workers do not spend their whole working week at home but spend some time at a central office or at client's premises (Kinsman 1987). The number of such workers is growing rapidly, in the USA the number of tele-commuters (including those working at neighbourhood community centres) is estimated at 5.5 million rising at over 20% a year (Markby 1991), and in the UK is expected to increase rapidly in future years (NEDO 1989). The pattern varies somewhat from country to country and is strongly related to cultural styles of management and other economic and geographical issues. In the USA one major factor has been the environmental costs of transport to and from work and the dollar cost of building sufficient car parking.

The individual business operating from home is not the main concern here. Instead, it is the problems of social and managerial contact when the individual tele-workers are part of a team. At present the main obstacles to effective tele-working are seen to be managerial, not technological. However, it is the author's belief that this will become an issue as the individualised nature of the tele-worker's systems jars against the tide of integration.

### 2.3. Other groups, slow email

Workers in remote (but major) sites may experience similar problems to mobile and tele-workers. Although the promise of ISDN is a world of pervasive wide-band communication, it is likely to be some time before this is universally prevalent. Even where ISDN is available line costs will encourage intermittent use, not permanent connection as with a within-office network. These sites may well be connected over networks (such as Janet or Internet) or by occasional dial-up connections (such as UNIX's uucp), the principal means of computer based communication being email.

### 2.4. Individuation and integration

The common thread between these workers is that they work on potentially cooperative tasks and yet for protracted periods work without communication. However, the ethos of computing during the past few years (in addition to notebook PCs) has been integration: networking and groupware. These workers sit at the bifurcation point between these two streams of individuation and integration. From a CSCW perspective the question is can there be cooperation without communication?

## 3. Augmenting the time/space matrix

The above workers are characterised by their infrequent communications – especially the intermittent communication between their computing equipment. Their situation differs dramatically from, say, workers at different sites but linked by constant wide-band communications such as ISDN. Although the latter are distributed geographically, they are logically local so far as their computing is concerned. The standard time/space matrix does not make these distinctions clear.

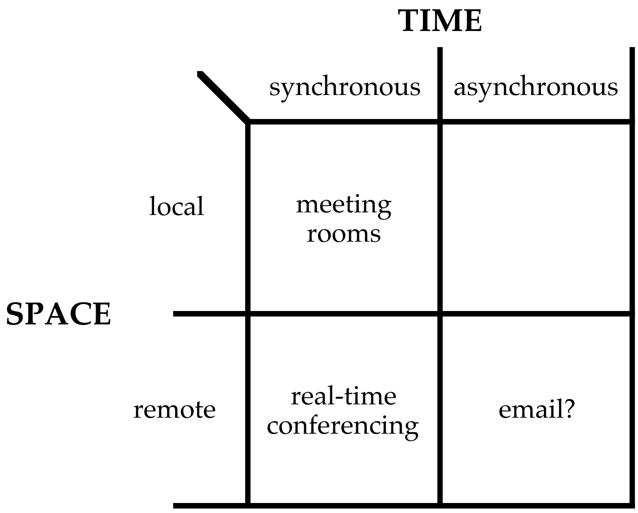



*figure 1.* standard time/space matrix

Imagine two people using an electronic conferencing facility over a wide-band network, perhaps even with a real-time video link. This communication would rightly be placed in the remote-synchronous square of the matrix. On the other hand, email is usually placed in the remote-asynchronous quarter. However, the two people could just as easily been using email to talk to each other, and been using their respective systems at the same moment. That is they were operating at the same time – remote-synchronous interaction? Clearly, from the participants' point of view the fact that they are working at the same time makes *no difference* to their interaction, the intermittent nature of the email communications dominates – they are working concurrently but are *unsynchronised*. The defining feature of the workers described in the previous section is not so much their physical remoteness, but the unsynchronised nature of their computer communications.

We will extend the time/space matrix substituting the single synchronous/asynchronous binary distinction with two more binary distinctions. One is concurrent vs. non-concurrent – literally whether or not people are working at the same time. The other is synchronised vs. non-synchronised, which is based on the level of communication between the participants' *computer* equipment. Synchronised means that the rate of communication is fast compared to the normal units of work (say for most purposes less than a few seconds) whereas unsynchronised means that the workers may well work for significant periods of time without their computers communicating. For the workers described above, this period varies from a few hours to several days.

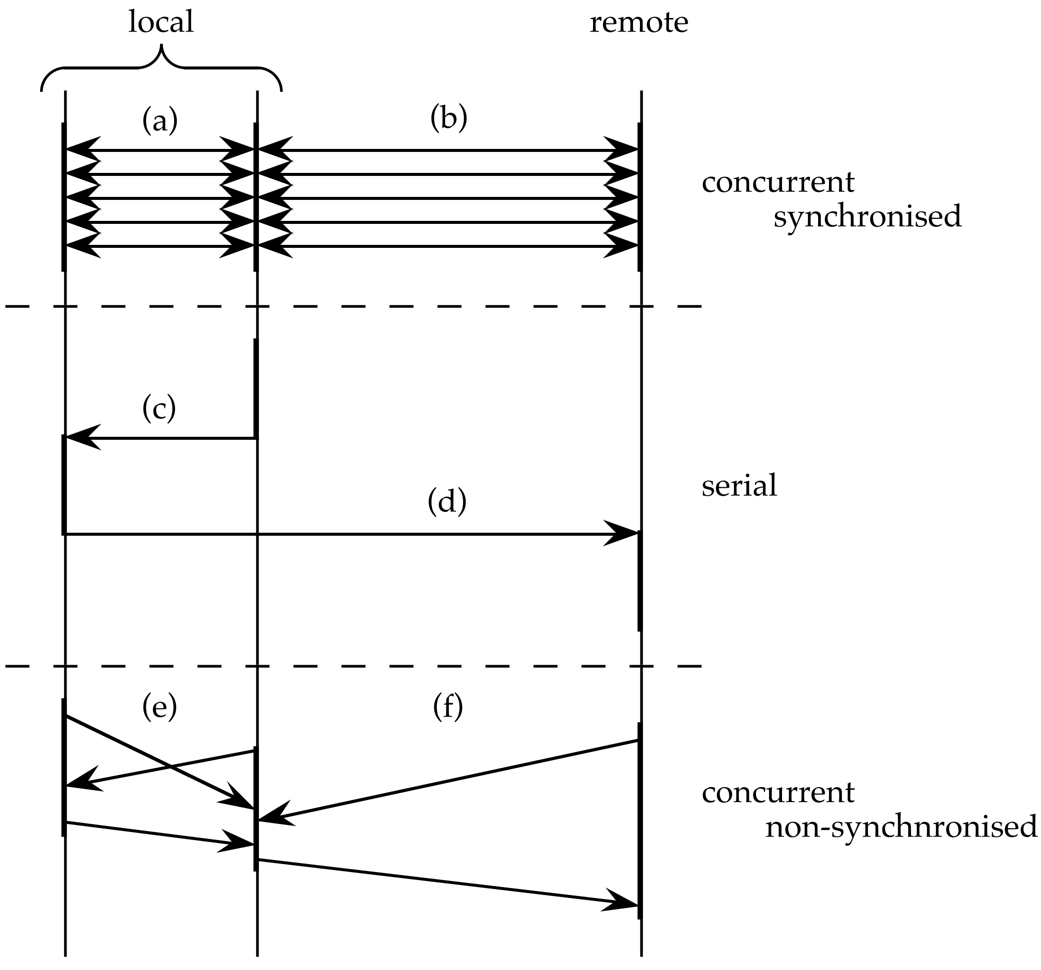



*figure 2.* synchronised and unsynchronised modes of work

The basic distinction is between synchronised/non-synchronised and concurrent/non-concurrent. However, for unsynchronised systems there is little difference between concurrent and non-concurrent working (as is the case with email). We thus end up with a ternary distinction: concurrent synchronised vs. serial (non-concurrent synchronised) vs. (possibly) concurrent unsynchronised. The modes of work are demonstrated graphically in figure 2, and we will discuss each in turn. The diagram represents timelines for three users, two at the same site and one remote. Time runs down the page and the bold vertical lines represent periods of each user's activity. Finally, the arrows denote communication between the user's workstations.

(a)&(b) concurrent synchronised local and remote – Working at the same time with tightly coupled workstations. These correspond to the standard categories of synchronous co-present and remote. Examples of the former (a) include meeting rooms such as Colab (Stefik et al. 1987) or shared editors such as ShrEdit (CSMIL 1989). The latter (b) includes video and desktop conferencing (Ishii and Miyake 1991) which aim to restore some of the sense of engagement of face-to-face communication.

(c)&(d) serial local and remote – In these cases the participants are working at different times, but their shared information is synchronised. This would occur for instance with a dial in bullitin board (with for the sake of argument only one dial-in line!). The information one sees is always up-to-date, but there is no concurrent access. A similar local example would be two draftsmen working on a drawing using a single workstation, who, say, job share so that they do not occupy the office at the same time, but nevertheless cooperate on the task.

The major problem in such a situation is maintaining context. The reasons for performing some action may not be apparent to another participant coming to the shared data later. In concurrent synchronised interaction such problems are likely to be ironed out as they occur, however, in serial interaction there is only as much context as the previous participant *explicitly* recorded.

(e) concurrent non-synchronised co-present – For example, workers in the same office with non-networked machines. They are working at the same time on common tasks, within earshot of one another, but their machines are not as closely coupled as they are. The major difference between these workers and the final group (f) is that they may use direct communication to resolve some of the problems that arise.

(f) concurrent non-synchronised remote – This is the category probably meant by asynchronous/remote in the standard matrix and is where the highly distributed workers, which are the subject of this paper, sit. The communication is intermittent, and communications may even 'cross' as do, for instance, email messages or ordinary paper letters. They obviously have the same problems of loss of context as serial work (c) or (d). In addition, they may experience considerable problems coordinating their activity. As well as the obvious problems due to crossed mailings, they may work on the same objects at the same time creating multiple versions and thus causing future problems in resynchronising data.

In the standard time/space matrix it is most likely that a system described as local/asynchronous would sit in category (c) above whilest one described as remote/asynchronous would sit in category (f). This demonstrates the conflating of remoteness with synchronisation in the standard framework.

There are a few overlaps between the areas described above. The examples of serialised communication (c) and (d) were somewhat contrived in order to reinforce the distinction. The examples given can be thought of as necessarily serialised, where there is no possibility of more than one person working at once. This may sometimes be forced by the software, for instance by the use of a strict locking protocol on shared objects. The systems are synchronised and the workers may even be concurrently active on different tasks, but for any particular shared object only one participant is active.

In other circumstances the serial access may be accidental. The workers just happen to arrive at different times. Such a system may then support both (a/b) and (c/d) type interaction, and even mixtures of the two. Figure 3 shows an example of such interaction. The design rational for such a synchronous/asynchronous group editor is described in (Miles et al. 1993). The important thing is that whenever participants are acting concurrently their activities are synchronised – there is an illusion, possibly with a distributed archirecture over a wide area, of a single information server.

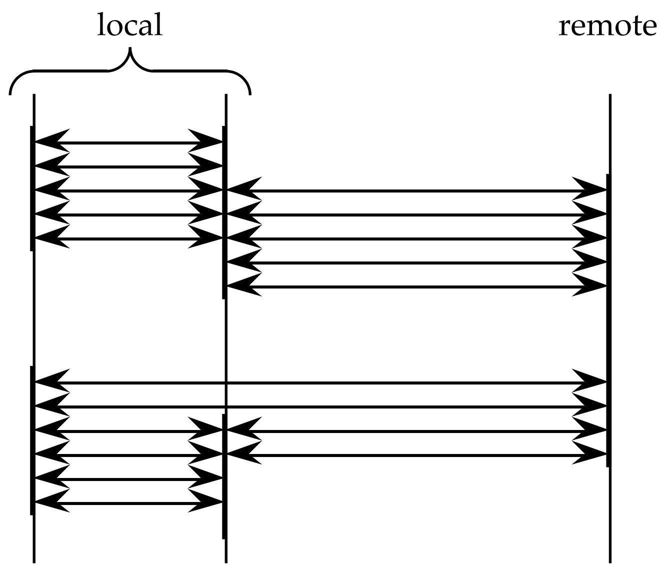



*figure 3.* mixed serial/concurrent synchronised work

It is worth emphasising the importance of timescale in interpretting figure 2. Almost all distributed applications look somewhat like (e) or (f) at a fine enough timescale. However, if the timescale is small enough the systems may be effectively synchronised from the point of view of the participants. For instance, Ellis and Gibbs produce diagrams rather like (f) to describe the communications between workstations in their distributed group editor GROVE (Ellis and Gibbs 1989). However, they use clever algorithms which ensure that these communications delays are not apparent to the user. Just what timescale constitutes 'apparent' depends on the modality of communication, the Ellis and Gibbs scheme could probably handle delays of several seconds for shared editing, and in our own experiments at York the determining factor in text based conferencing was the participants' composition time not the transmission time (McCarthy et al. 1993). However, in trans-Atlantic telephone calls (and more dramatically in video-conferences) a delay of a few seconds can be very disconcerting and cannot be hidden by software tricks!

As the final category non-synchronised working, is our primary concern, we will discus the various sub-divisions in detail in the following section.

## 4. Different forms of non-synchronised working

Many of the issues described apply to all distributed workers, but there a few further sub-divisions which make an occasional difference to the support systems we design. To some extent the diagrams (e) and (f) in figure 2 show the worst possible scenario: the communications took time to get from one site to another and could cross each other during transmission. This situation would arise if one were using email as a transport medium (either for inter-personal or inter-computer communications).

Obviously all media have some lag in them, but many can be regarded as effectively instantaneous. The use of such media does not necessarily mean that the interaction is synchronised as the communications may still be intermittent. This situation is depicted in figure 4. The participants are largely working independantly, but occasionally their workstations communicate in a synchronised manner. Examples of this would be when a mobile computer is brought back into the office and it and the office machine are brought up to date with one another, or when two machines make contact via a modem. At a human-human level we get parallel communications scenarios when workers meet after working alone, or when they telephone one another. Various constraints may mean permanent synchronised communication is impossible or impractical. This may be because of physical distance – the mobile worker is not in the office, or because of cost – a permanent phone call/modem link may be prohibitively expensive.

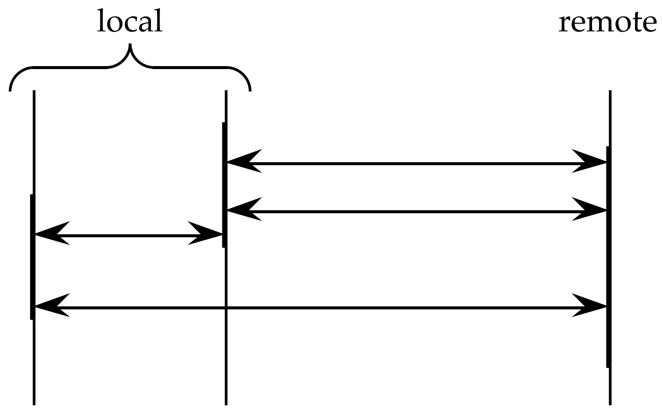



*figure 4.* intermittent synchronous communication

For the participants themselves the main impact may be in terms of direct communication. We all know the enormous difference even occasional face-to-face meetings can have. For the computer-computer communications the difference between figure 2(f) and figure 4 is less crucial. It certainly makes the algorithms easier to implement if one can assume occasional bidirectional interaction between machines. However, the software should be able to make the difference between the two negligible for the user.

Another, possibly more important, distinction is whether the occasional communications happen outside the user's control, or whether they are *elective*. In the case of elective communication, two users (say Jack and Jill) may contact each other and Jill may decide that she would like to see an up-to-date version of Jack's document. She could then choose to establish a temporary computer-computer link (they may both simply plug in their modems) and obtain the necessary information.

In human-human communication the telephone is, depending on the availability of the callee, a pervasive elective communication medium. Even mobile workers may have a portable phone (if they can afford to use it!).

In much of the preceding discussion the distinction between direct human-human communication and the state of the participants' shared data has been important. The participants observe one another's actions on the shared artifacts – both outcome and process. This indirect channel of communication *through the artifact* (Dix 1994; Miles et al. 1993) can be as important as direct communication, perhaps more important as it is so closely tied to the task. Unfortunately, it is exactly this communication through the artifact which is most weakened by distributed working.

The framework in (Dix 1994) emphaises the importance of integration between direct communication and the task artifacts. The direct conversation is about the task, and hence refers to the artifacts (deixis). Furthermore, the participants are aiming towards a shared understanding of the cooperative task (Clark and Brennan 1991). However, as mentioned earlier, the conversation of the participants can only be fully understood in the light of the current state of the task objects and an individual's manipulations of the shared objects often requires commentary to be understood by the other participants. Thus, even where there are existing direct communication channels, there will be need for additional direct channels implemented in the computer system, but tied within the context of and able to refer to the task objects themselves.

## 5. Issues for CSCW

This section describes some specific issues for the design of systems supporting cooperative distributed work. The list is, I'm sure not complete, but the purpose of this paper is to inspire more work in investigating the area. In particular, the issues described are not the result of any extensive requirements analysis, but are based mainly on the author's own experience and what can be gleaned from related literature. Each is self-evidently important, but a more systematic analysis of the work patterns of these workers may well uncover other broad areas, or more detailed issues with regard to those below.

The discussion below centers on three areas:

- direct communication – establishing context and deictic reference.
- indirect communication – in particular resynchronising shared data after concurrent updates.
- externalisation – the positive effect that distributed work may have on groupware products.

### 5.1. Direct communication

We have already discussed many of the specific problems for direct communication. Many of these are problems are common to serial and unsynchronised interaction. The problems of the former are discussed in detail in (Miles et al. 1993) and several existing mechanisms are reviewed (although serial communication is described there as asynchronous in keeping with the standard CSCW nomenclature). The emphasis will therefore be on adapting the conclusions in that paper, aimed at serialised communication, to take into account the specific needs of unsynchronised working. The problems of direct communication can be summarised as re-establishing context.

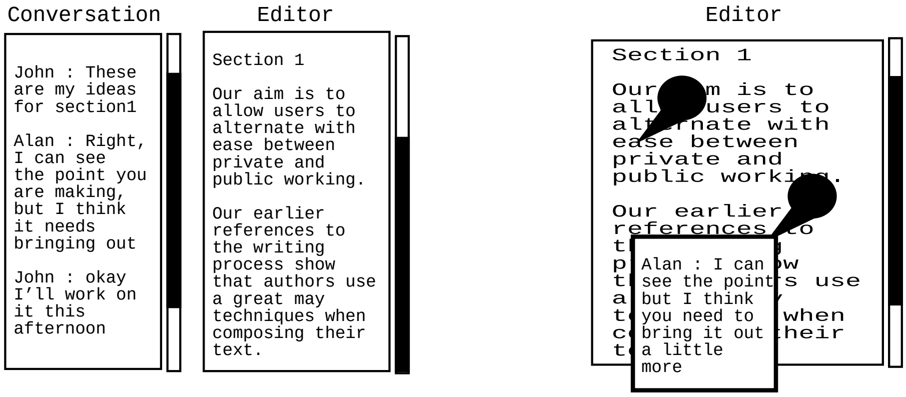
To some extent one might ask whether there is any problem with direct communication as the most successful groupware tool is surely email. This is an unsyncronised communication medium and is well suited to distributed work. Furthermore, there is already a large corpus of practical and theoretical work addressed at email communications. For instance, there have been many 'value added' email systems: intelligent filters such as the LENS system (Malone et al. 1987) and Mailtrays (Rodden and Sommerville 1991), or role-structured mail such as in Cosmos (Dollimore and Wilbur 1991) or coordinator (Winograd and Flores 1986).

If email is to be a useful medium for distributed workers it should be pervasive – available at all times, not just when connected to a central facility. In theory there is no problem, it is easy to imagine 'sending' an email whilest on a train which is then stored on the portable computer until it is next connected to a network. Similarly, a home computer could store up messages ready to be transfered by floppy disc or by dialing into a central computer during off-peak phone hours. Such features are surprisingly rare.

So, messages must clearly be able to be produced in the context of the sender's work. However, it must be possible for the recipient to re-establish that context – without support the burden for this rests entirely on the sender's prose! As noted previously, an important part of communication about a task is the references to the task objects (deixis). Any messaging system should allow easy reference to shared objects. Remember that the shared objects may even have different names at different sites, so even including textual names causes considerable difficulty. Some centralised mail systems include such facilities. For instance, several commercial systems allow 'enclosures' not really references, but a start. Quilt (Leland et al. 1988) does better as it allows references to parts of the Quilt shared hypertext to be placed in email messages. These links can then be followed by the recipient. Allowing such deictic links in a highly distributed environment will mean that the direct communication medium must be closely tied to the means of supporting shared data.

Conferencing systems often add more structure than simple email in order to overcome some of the disjointness of email exchanges. This structure is largely of two forms, either linear, with contributions on a topic being appended to the transcript, or a hypertext. In our design for a group editing environment (Miles et al. 1993) we chose to use a linear form because "...traditional linear text provides a continuous, unwinding thread of context as ideas are proposed and discussed" (Conklin and Begeman 1989) and was thus better suited for asynchronous users catching up with a previous synchronous session than a hypertext. However, it is only possible to maintain a linear conversation space in a synchronised environment. One possibility would be to have 'nearly linear' hypertexts associated with each topic, where any synchronous computer based conversation is linearised, but concurrent conversations occupy different frames – the structure of the hypertext reflecting the structure of the computer sessions which generated them.

We discussed above the importance of allowing refernences to task objects from the conversation, another approach to maintaining context is to embed the conversation into the task objects themselves. For instance, in our group editor each section of a document has an associated text transcript. By associating conversations with their structural context, it is more likely that one can obtain a largely linear structure during concurrent activity. Another, finer grain option, is to associate annotations with specific points in the objects. The shared data then itself begins to act as a form of hypertext revealing the context of its creation. Both options are illustrated in figure 5. Annotations are particularly common in co-authoring systems as for example in Quilt (already a hypertext) and the PREP editor (Neuwirth et al. 1990).




*(a) A Segment Editor with a conversation space*　　*(b) A Segment Editor with annotations*

*figure 5.* embedding conversation in context

Finally, one may want a more dynamic notion of context. This may be partially supplied by voice annotations, but a more sophisticated system could include animated messages, such as the active messages in the Andrew Message System (Boronstein and Thyberg 1988), or replays of the context in which an object was created. In general, one wants to recreate some of the feeling of 'the moment' for the reader. Obviously, the more distributed and asynchronous the work, the better the tools we must supply in order to restore the participants' sense of engagement.

### 5.2. Re-synchronisation of shared data

The specific problems of handling shared data objects is discussed in detail in (Dix and Miles 1992) and this sub-section is an overview of the arguments presented there. In a centralised or tightly knit architecture there are many options for regulating access to shared data resources. These range from simple file level locking, through relatively fine grain locking of objects or document sections to tightly coupled group editors, such as Grove or ShrEdit. Also various social protocols, roles and ownership rights can be used (Leland et al. 1988).

For highly distributed systems the options are much reduced. Obviously fine level sharing and locking are nor viable alternatives. Larger level locking and protocols such as baton passing are possible, but the time scales of periods of locking will of course be limitted to the frequency of the communications. For mobile workers or some home working, where communication is only established, say, at most daily, the level of pre-planning involved is immense. Obviously, many joint documents are prepared in just such a fashion, exchanging drafts via email, or more traditionally by passing of paper copies. However, the thrust of recent work in groupware has been to allow, where the participants desire it, finer grained working. For example, I am, at the moment, writing another paper with a colleague.

My colleague 'has the baton' so to speak, and is working on the paper – on a notebook computer whilest at a meeting in France. I have also been looking through the paper and noticed various spelling and minor mistakes in the paper copy. However, I must wait until I get the baton back in order to make these changes. Obviously, systems allowing online annotations would make this task easier as I could annotate a version now and then my colleague or I could perform the changes when we next meet.

For more major changes such annotations are less effective. In a traditional software tools environment, I might take my electronic copy of the document and amend it, then when we next met, we could compare the versions by eye or using a difference tool (such as UNIX 'diff'), and merge them. This process is the re-synchronisation of data. It is clear that without rigid formal controls on shared data such diverging versions of shared objects are inevitable. However, in any creative endeavour the formal controls are likely to stifle spontaneous work. The solution is then to supply tools which both support the existence of such divergent versions and also supports the re-merging of those versions when the systems re-synchronise.

Few systems support this activity. The only one of which I know for highly distributed systems is Liveware (Witten et al. 1991) which is a distributed database built over HyperCard (Apple 1987). It is transfered from site to site using floppy disk transfer and uses the analogy of a computer virus for its transmission. When different versions of a Liveware database meet they automatically merge based on the timestamps of the data items. In order to make such automatic merging possible without user intervention, each item in the database has a single owner. Thus the most recent entry is always the correct one. Effectively this means that although the database as a whole is distributed and unsynchronised (case e/f from figure 2.) each individual record has serialised update (case c/d). Such a 'most recent is right' policy is not appropriate where different participants are allowed to update a shared document. In this case an automatic merge is only appropriate when the newer document has been derived directly from the older one. If the documents represent divergent developments some user intervention is usually necessary (although slightly more intellegent automatic merges may be possible). Supporting the users in this re-synchronisation process demands not only highlighting of differences, but also a comparison with the original source of both documents. In addition, this is just where additional commentary, direct communication explaining the changes, is useful. Harrison *et al.* describe a scheme, primarily for managing software units, which restricts automatic merges, to just those 'safe merges' where one version is derived from another (Harrison et al. 1990). They allow users to break locks (they call this a *lenient locking protocol*) which then introduces the possibility of divergence. They then restrict remerging to the safe merges, but do not support the users further in the re-synchronisation process.

We have proposed a scheme called multiple source control or MSC (Dix and Miles 1992) which extends the concepts of standard branched version control systems, such as RCS (Tichy 1985), to also allow remerging and other features necessary for distributed working. MSC is based upon replicated version trees. For each file, document or section of a document there is a branched version tree. Each version belongs to one or more *threads*. These threads are linear streams corresponding to the sequence of updates for each site or user (and possibly additional streams such as the 'official' draft of a document). The threads are like standard version tree branches, but are branches with the *intention* of remerging. Thus the version tree for each document would look rather like a braided cord with versions constantly branching and remerging. Because the whole version tree is stored (using differences for efficiency), whenever two versions are merged the system can check whether one is derived from the other, and if not can extract the most recent common ancestor. This common source will be invaluable in handling the subsequent user-controlled merge. The use of dynamic pointers (Dix 1991; Dix 1995b) will also aid this merge process as they allow the system to easily relate locations in divergent versions and in their common source which, in particular, allows annotations made in one version to be viewed as part of a parallel version.

### 5.3. Externalisation

So far the paper has been concerned with problems with cooperative work for distributed workers. However, to finish on a positive note, this penultimate section shows that the rigours of distributed working can be an opportunity for the introduction of generally effective groupware systems.

The storage and computation power of a computer system can be seen as an externalisation of aspects of the user's own cognition. Some of this externalisation is explicit (e.g. the contents of an address book), but most of the procedural and causal aspects (why those particular addresses?) is implicit. Groupworking also demands an externalisation of the members own thinking in order to establish mutual cooperation. In face-to-face working this externalisation is often itself implicit in the joint activity and subtle social cues between the participants. However, where the social constraints are reduced one must often work harder to present this information to one's colleagues.

Individual distributed workers will often experience breakdowns in their use of computerised data, which would not occur with a centralised system. This may happen because of data divergence and merging as discused above: for instance, I have recently modified a reference in the database of my home machine, but must remember to modify the copy on the office machine. The breakdowns may also occur due to locking or downloading data: normally the files required for a specific work-session emerge as a function of the work being performed, however, when working on a portable machine one must remember to download up-to-date versions of all files necessary during the next day or two. Only recently I forgot to write to floppy disc a document I had been working on at home and urgently needed – some I rewrote from scratch and some passages my wife dictated to me by telephone whilest she read from my home computer. Workers with such problems will be glad of tools which help them to organise their distributed information resources. These tools will by their very nature capture some of the same information which would be helpful to co-workers.

One way of looking at the single worker engaged in different sites is as two personae, one at each site. Cockburn and Thimbleby (Cockburn and Thimbleby 1991) call this a *reflexive* perspective on CSCW Thus, the individual worker experiences many of the problems of a group and thus will begin to work in ways more fitting for group cooperation. For instance, such workers may record information about what they are doing and why in order to help themselves as they return to the task at a remote site. This is, of course, just the information required by their colleagues.

It is widely acknowledged that groupware is more talked about than used. Various reasons have been suggested: those who benefit may not be those who have to put in the effort, the free rider problem, the lack of immediate benefits for each individual. One of the chief problems is that a groupware system is not useful until used by a significant group ... and is not used until useful: "a critical mass of users is essential for the success of any communication system" (Eurlich 1987). Grudin in his analysis of the failure of CSCW systems looks at these and other factors in the light of various case examples (Grudin 1988). One of these is the electronic calendar, a frequent example of an apparently useful groupware system. People simply do not keep their electronic diaries up-to-date. Do people simply dislike the electronic medium? Strangely enough personal organisers are very sucessful and their users do keep their *personal* electronic diary up-to-date. If this existing store of information can be recruited into a groupware system by periodically 'docking' the personal system (Beard et al. 1990) then one could almost overnight obtain a critical mass of users. There are, of course, other problems to face, such as maintaining privacy, but one crucial barrier, getting information online, has been broken.

## 6. Summary and much future work!

There is an identifyable group of workers, whom I call distributed workers, who share a common set of problems due to their infrequent communications with each other and shared data. The standard time/space matrix does not readily identify their specific problems so this has been extended to distinguish cuncurrency from synchronisation. Doing so we found that the term asynchronous as used in CSCW circles, covered two classes of interaction with different requirements and problems. Some specific areas have been discussed concerned with establishing context in computer-supported direct communication and the resynchronisation of concurrently-updated shared data. We also saw that the problems faced by individual distributed workers may in fact lead to them adopting practices which can be used to enhance groupware systems. The final summary normally stresses the advances described in the paper. However, the intention of this paper is not so much to present specific solutions to the problems of distributed workers, but to encourage others to apply their theories and products to this growing market.

## References

Apple (1987). *HyperCard User's Guide*. Apple Computer Inc., Cupertino, CA.

D. Beard, M. Palaniappan, A. Humm, D. Banks, A. Nair and Y.-P. Shan (1990). A visual calendar for scheduling group meetings. *CSCW'90 Proceedings of the Conference on Computer-Supported Cooperative Work*, ACM SIGCHI & SIGOIS, 279–291.

N. S. Boronstein and C. A. Thyberg (1988). Cooperative Work in the Andrew Message System. *CSCW'88 Proceedings of the Conference on Computer-Supported Cooperative Work* , ACM SIGCHI & SIGOIS. 306-323.

H. H. Clark and S. E. Brennan (1991). Grounding in Communication. *Socially Shared Cognition* , Ed. J. L. a. S. D. B. L.B. Resnick.

A. J. G. Cockburn and H. Thimbleby (1991). *A Reflexive Perspective of CSCW*. Stirling University, Scotland.

J. Conklin and M. L. Begeman (1989). gIBIS: A tool for all reasons. *Journal of the American Society for Information Science,* (March 1989).

CSMIL (1989). *ShrEdit: A Multi-user Shared Text Editor: Users Manual*. Cognitive Science and Machine Intelligence Laboratory, The University of Michigan.

A. Dix, J. Finlay, G. Abowd and R. Beale (1993). *Human-Computer Interaction*. Prentice Hall.

A. J. Dix (1991). *Formal Methods for Interactive Systems*. Academic Press.

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

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

A. J. Dix (1995b). Dynamic pointers and threads. *Collaborative Computing,* **1**(3): 191–216.

A. J. Dix and R. Beale (1995). Information requirements of distributed workers. *Remote cooperation: CSCW issues for mobile and tele-workers* , Eds. A. J. Dix and R. Beale. Springer Verlag.

A. J. Dix and V. C. Miles (1992). *Version control for asynchronous group work*. YCS 181, Department of Computer Science, University of York, (Poster presentation HCI'92: People and Computers VII).

J. Dollimore and S. Wilbur (1991). Experiences in building a configurable CSCW system. *Studies in Computer Supported Cooperative Work: Theory, Practice and Design* , Ed. J. M. B. a. S. D. Benford. North-Holland. 173-181.

C. A. Ellis and S. J. Gibbs (1989). Concurrency control in groupware systems. *Proceedings of 1989 ACM SIGMOD International Conference on Management of Data, SIGMOD Record,* **18**(2): 399-407.

S. F. Eurlich (1987). Strategies for encouraging successful adoption of office communication systems. *ACM TOOIS,* **5**: 340-357.

J. Grudin (1988). Why CSCW applications fail: problems in the design and evaluation of organisational interfaces. *CSCW'88 Proceedings of the Conference on Computer-Supported Cooperative Work* , ACM SIGCHI & SIGOIS. 85-89.

W. H. Harrison, H. Ossher and P. F. Sweeney (1990). Coordinating concurrent development. *CSCW'90 Proceedings of the Conference on Computer-Supported Cooperative Work,* ACM SIGCHI & SIGOIS, 157-168.

H. Ishii and N. Miyake (1991). Towards an open shared workspace: computer and video fusion approach of TeamWorkStation. *Communications of the ACM,* **34**(12): 37-50.

F. Kinsman (1987). *The Telecommuters*. Wiley.

M. D. P. Leland, R. S. Fish and K. Robert E (1988). Collaborative document production using Quilt. *Proceedings of CSCW'88* , . 206-215.

T. W. Malone, K. R. Grant, K. Lai, R. Rao and D. Rosenblitt (1987). Semistructured messages are surprisingly useful for computer supported coordination. *ACM Transactions on Office Information Systems,* **5**(2): 115-131.

D. E. Markby (1991). Commercial Reality - Experience from the United States. *Teleworking - Real Business Benefits*, report of BCS/IEE seminar,

J. C. McCarthy, V. C. Miles, A. F. Monk, M. D. Harrison, A. J. Dix and P. C. Wright (1993). Text-based on-line conferencing: a conceptual and empirical analysis using a minimal prototype. *Human-Computer Interaction*, **8**(2).

V. C. Miles, J. C. McCarthy, A. J. Dix, M. D. Harrison and A. F. Monk (1993). Exploring designs for a synchronous-asynchronous group editing environment. *Computer Supported Collaborative Writing* , Ed. M. Sharples. Springer-Verlag.

NEDO (1989). *Working by Wire*. National Economic Development Office.

C. M. Neuwirth, D. S. Kaufer, R. Chandhok and J. H. Morris (1990). Issues in the design of computer support for co-authoring and commenting. *CSCW'90 Proceedings of the Conference on Computer-Supported Cooperative Work* , . ACM SIGCHI & SIGOIS. 183-195.

T. Rodden and I. Sommerville (1991). Building conversations using Mailtrays. *Studies in Computer Supported Cooperative Work: Theory, Practice and Design* , Ed. J. M. B. a. S. D. Benford. North-Holland. 159-172.

M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel and D. C. Steere (1990). Coda: A Highly Available File System for a Distributed Workstation Environment. *IEEE Transactions on Computers*, **39**(4): 447-459.

M. Stefik, G. Foster, D. F. Bobrow, K. Kahn, S. Lanning and L. Suchman (1987). Beyond the chalkboard: computer suuport for collaboration and problem solving in meetings. *CACM*, **30**(1): 32-47.

W. F. Tichy (1985). RCS – a system for version control. *Software Practice and Experience*, **15**(7): 637–654.

T. Winograd and F. Flores (1986). *Understanding computers and cognition : a new foundation for design*. New York, Addison-Wesley Publishing Company, Inc.

I. H. Witten, H. W. Thimbleby, G. Coulouris and S. Greenberg (1991). Liveware: a new approach to sharing data in social networks. *Int. J. Man-Machine Studies*, **34**: 337-348.