Developing a tool to support collaborative dialogues and graphical representation of ideas

M.B. Twidale, T. Rodden, I. Sommerville

Computing Department, Lancaster University, Lancaster LA1 4YR.
Email: mbt@uk.ac.lancs.comp

1 Introduction

The aim of this paper is twofold: firstly to describe the Designers' NotePad (DNP) and its potential as an advanced form of Computer Mediated Communication (CMC) for Distance Education, and secondly to describe the techniques used in the development of the DNP that may be helpful in the development of other CMC systems. With regard to the former, we believe that the DNP offers considerable potential in supporting collaborative dialogues by its use of 'graphical text', i.e. short text fragments positioned two dimensionally and supported by graphical notations such as various kinds of links, surrounding shapes and the use of colour. Using the system, idea structures can be developed and refined with great ease and speed and used as the basis of a discussion for further developing the ideas.

With regard to the experiences and techniques used, we wish to emphasise the substantial influence of the interface on usability and effectiveness of the system. This effect can completely swamp the effects of other carefully prepared features of the system to support learning. Therefore we advocate devoting considerable resources to the iterative design and evolving refinement of an interface that is easy to learn and easy to use.

2 Background

The Designers' Notepad was developed as part of a joint project between the Computing and Sociology departments at Lancaster University. The aim of the project is to investigate computer support for collaborative software design, in particular the early stages of the design process which existing CASE tools offer little support for. Part of the project involves undertaking an ethnographic study of the software design process in order to inform the development of computer tools which can better support this process (Sommerville et al. 1993). However ethnographic study and analysis is inevitably time-consuming and so it was necessary to develop a prototype system before the full results of the study emerged. Therefore a rapid prototyping approach was chosen to facilitate the incorporation of the results and recommendations of the study as they became available. The intention was to develop a simple to use core system that could be quickly tested on users. The experience of use then informed subsequent development.

3 Overview of the Designers' NotePad

Figure 1 shows the DNP interface. In order to make the learning of the system easy, there are a small set of core features which are sufficient to enable productive activity to take place in a short time. The basic unit is a design: rather a misnomer here, essentially it is a window into which idea elements can be placed. The design in the figure was used in preparing this paper.

The user creates an entity by typing in a design window. She may then move it with the mouse. Linking is done by selecting one entity and then clicking on the entity one wants to link to with the shift key down. We also provided Textnotes which are note pads based on the Post-It Note metaphor (see figure 1) and allow users to attach one or more notes to an entity. These can be used for more textual comments, ideas, opinions, paragraphs of a final document, references etc. A variety of Textnote types are provided and users may define their own (including form-like structures). An entity with Textnotes has an icon attached (eg. the entity 'small scale' in figure 1) and the notes can be examined by clicking on the icon. Designs can be saved and loaded from a file and a paper report may be created containing a screendump of the design and a list of the entities and their Textnotes.

Each entity may itself be expanded to become a subdesign. A new window is opened and entities and links can be created in the normal way. Subdesigns may contain entities that are themselves subdesigns. A loose form of typing for entities and links is provided using colour, shape and labels. The user controls the degree to which she wants to use this typing. The type of an entity or link can easily be changed at any time. The use of typing here is closer to the concept of styles in a wordprocessor than types in a programming language. Eventually a software designer may wish to

use more formal and rigorous typing in order to benefit from type checking, but that is appropriate for the later stages of software design; it too restrictive for the early stages.

Entities can be easily moved within and between designs. Additional features enable the design to be annotated. These include a framing facility for grouping related collections of entities. Groups can then be moved en masse to assist rearrangement. In order to control the complexity of an evolving design, a group can be 'pushed down' to form a new subdesign initially containing the members of the group and replacing the group in its originating design with a single named entity. In figure 2 the group of entities from figure 1 relating to requirements have been pushed down to form a subdesign, with a new entity 'System Requirements' replacing them in the parent design.

Figure 1. The DNP interface, showing a design used in the preparation of this talk.

Figure 2. The subdesign created by pushing down a group from the main design.

4 Educational Potential of DNP

The system was initially developed to support the early stages of design, which involves the

following activities:

- Brainstorming
- Ideas organising
- Refining
- Revising
- Rearranging
- Gradually adding detail

These are activities that occur during learning in many domains and so the DNP has far wider applicability as a tool for supporting learning activities. Furthermore, the development of a minimal core system for DNP as part of the rapid prototyping approach meant that the simplified features provided were necessarily not domain specific. Note that this claim for domain independence is unlike those normally made for domain-independent learning environments. These are generally complex systems where domain independence is due to a generic architecture that enables domain specific elements to be slotted in. For example they may include an expert system shell into which domain specific knowledge may be inserted. Our system's generality is due to the nature of the task to be supported (early idea organisation) and the simplicity of the core features. Also our focus is on the provision of appropriate tools to support users rather than the attempt to automate certain activities.

Thus the DNP can be regarded as an educational tool supporting concept mapping and other techniques for developing and refining ideas. Its key advantage is ease of learning and use. Vague, semi-formed ideas can be entered quickly and easily. If the user believes that all entries are easy to modify and correct, she need not concern herself initially with their ultimate form. This can prevent the block of premature commitment (Shipman & Marshall 1993). If she is free from commitment to form, she can begin by focussing on the broader issues. This inevitably involves some ambiguity, but the time to disambiguate is later on in the process. There are two reasons for the existence of ambiguity: firstly, in order to establish any form of overview, it is necessary to make gross approximations, 'broad brush strokes' are a useful metaphor. Secondly, in cases both of design and learning, the user is not merely keying into the system a pre-existing mental representation. The structure evolves as a result of its construction. So subtleties gradually emerge and are handled. As a map grows it eventually becomes clear that some terms are ambiguous and need to be refined (contrast this with the first reason where ambiguous terms are deliberately chosen).

5 Rapid Prototyping

During the early stages of software design, designers are particularly creative and necessarily handle ambiguous ideas. Little is known about the nature of this activity and consequently how best to support it. The same is true of learning activities involving creative idea organisation, and collaborative learning in general. Asking the users for requirements for such a system may be of little use even if they are experts in the activity. It is a notorious problem in the twin fields of requirements capture (Sommerville 1992) and knowledge acquisition (McGraw & Harbison-Briggs 1989) that experts have difficulty in articulating what they are actually doing, let alone what they would like from a system in order to do it better.

For both these reasons, the process of rapid prototyping for systems development is particularly appropriate. Our approach was to develop a very simple core system (principally; boxes, links, subdesigns and textnotes) and then to observe its use in authentic tasks. Although users find it difficult to articulate new requirements to support their areas of expertise, they can be most effective (and vociferous) in saying what is wrong or missing from a provided system or prototype.

In all our studies we were concerned to test the system on authentic tasks. That is, we asked our volunteers to bring along a piece of work (such as preparing a design, talk or paper) that they would have to do anyway. We asked them to try using DNP until they felt that it was no longer of use to them in solving their problem. We observed them using the DNP to work on this problem, noting their patterns of activity, what they found difficult and the features they would need to have added to DNP in order to progress further. After the study we discussed informally with the volunteers their experiences of using the system and their comments on improvements.

The advantages of using authentic tasks rather than a uniform task of our own invention that we present to all volunteers are:

- It is naturally much more engaging for the user.
- They ideally focus on their problem rather than our system, which is how the final system will be used. If the interface is clumsy and fails to allow this focussing we can spot what needs correcting.
- Real world problems are different from artificial tasks: they are messy, do not fit into neat

classifications, always contain exceptional features, are open-ended, contain elements of ambiguity and have a history and a future (so the system must cope with existing documents, ideas etc. and provide some product that can be used in later design stages or activities).
- There are inevitably implicit assumptions about the nature and style of use (and about the user, task etc.) in the design of the tool. In making up test problems, developers are in danger of incorporating these same assumptions. Thus the study will fail to reveal them. Outsiders' problems need not contain such assumptions and so serve as a better test.

This approach to development draws heavily on the work in participatory design; and approach to general computer systems development involving close participation with end users (Ehn 1989). Informal testing is cheap and relatively easy to arrange, but it should be begun as early as possible in the design stage and be repeated frequently throughout the continuing evolution of the system.

6 Gradualist Development

A consequence of the prototyping approach is that improvements to the system will be incremental. This, coupled with the need for a very simple core system with which to begin the testing cycle, implies a need for a gradualist strategy for testing and developing the functionality of the system. That is, we began by studying the use by individuals and small groups of users simultaneously working on the core system at a single workstation. This revealed the grosser interface errors and the improvements immediately needed for enabling progress with such activities. The changes were incorporated and re-tested.

For the early testing of the system it is quite in order for the developers to test it themselves. This might seem somewhat futile, but provided the activity is authentic (i.e. independent of the need merely to test the system) it can yield valuable results in a very rapid and economical way. For example, we used the DNP to plan the early stages of talks, reports and papers. Testing on developers is acceptable at the early stages because we are seeking to detect failure, not success. We can confidently predict that any feature that *developers* find difficult or clumsy to use will also be at least as awkward for normal users. In all cases interface failures 'scale out' to more complex contexts, although there is no guarantee that successes will. In the process of scaling out, new features and functionalities will certainly need to be provided, but there seems little point in worrying about these until the more basic problems detected in the simpler cases have been remedied.

Later testing of the system involved a process of 'pushing out' its coverage to more general contexts. This includes different activity types brought along by volunteers and different user types who will share fewer implicit assumptions about the DNP. So far we have had the following kinds of volunteer: system developers, people involved in the project but not directly involved in its development, postgraduate computing students, colleagues from Sociology, visiting academics, undergraduate computer scientists and undergraduates from other disciplines. The tasks that users have used it for include software design, project planning, organising research ideas and preparing essays, papers, reports and talks. The DNP has now been installed at a number of other sites and we are waiting to hear comments and criticisms from users who may not share our assumptions about what a tool must provide to be effective. We have also pushed out the context of usage by studying cases where the same groups and individuals continued to use the system over a number of design sessions separated by up to a week.

In all cases, observations of practice lead to requirements to better support the particular activity. In addition it may yield further requirements for the simpler cases considered earlier. Only when the most glaring errors and omissions are identified and remedied is it possible to spot the more subtle ones that are nonetheless crucial in their influence on usability. Similarly, it is easier to spot an issue in the simplest context in which it can arise, rather than always testing in the ultimate intended context which is usually an environment with so many issues that it hard to distinguish between them and their relative importance.

Figure 3 illustrates the idea of extending outward the base of testing. Local, cheap and easy to arrange tests on a basic system can catch the grosser errors and omissions. Only when these are eliminated can the more subtle ones become apparent. Some of these can also be detected by the same types of user. But others (to the right of the first dotted line) can only be detected by extending the user base and context of use. So we then extend testing to more diverse users who can catch even more subtle errors (but also others that our limited local testing could have detected but just happened to fail to do). Each type of user can detect errors and omissions to their left in the diagram, but generally will only be able to detect the more subtle of them after the others have been eliminated. This procedure can continue for any number of layers of increasing generality; the figure only shows three for simplicity. Each time the system is changed in the light of the testing, re-testing is necessary

both to find the next most subtle errors and omissions and to ensure that the change in the system has not introduced new problems. An additional benefit of this approach is that at all stages after the development of the core system, one has a working prototype that is usable by some subset of people for some subset of the intended contexts.

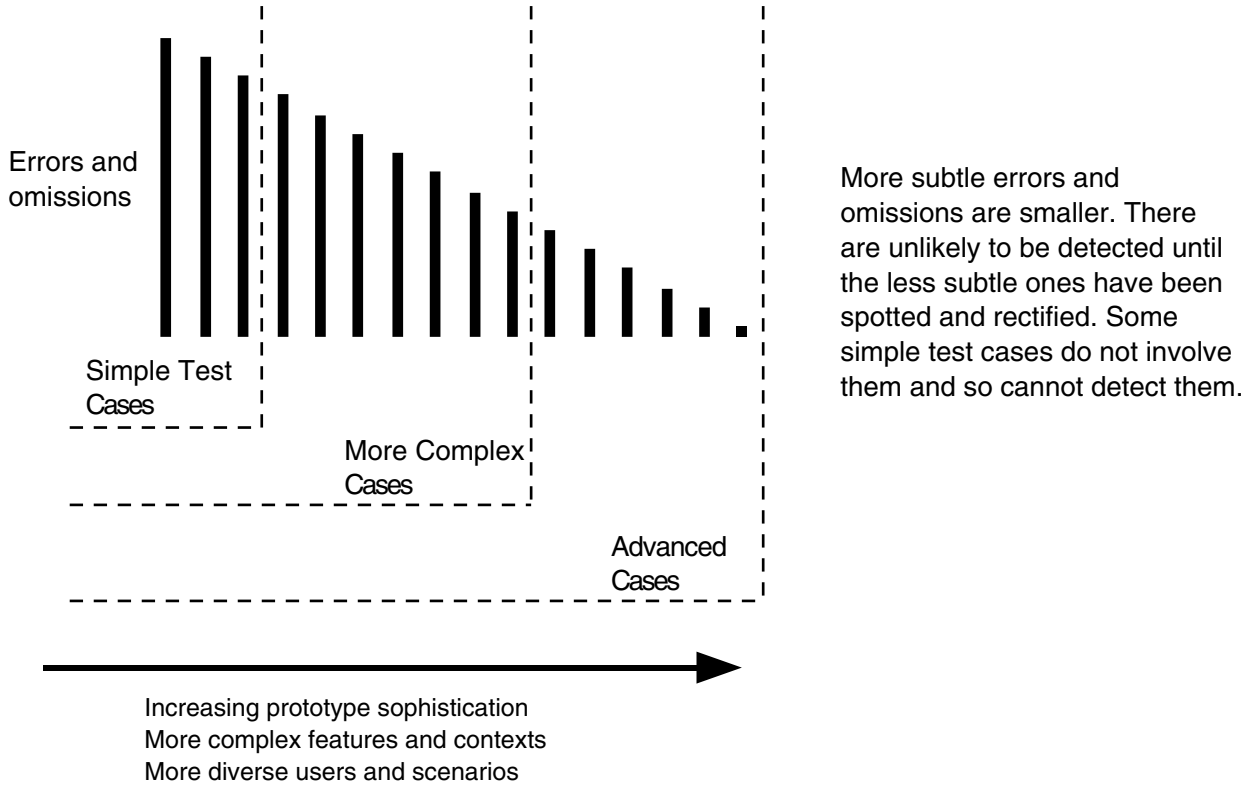

Figure 3. Illustration of the gradualist approach to developing and testing: using simpler cases first.

We next intend to 'push out' the coverage of the system to more complex contexts: asynchronous working by different group members on the same design, distributed but asynchronous working (where designers email the evolving design or its changes back and forth) and finally synchronous, distributed activities. Again the gradualist approach applies: problems that a single user has will scale out to multiple users at a single workstation, and the problems of the latter will scale out to cases of remote collaboration.

This approach is essentially complementary to the design work of for example Ishi & Arita (1991), Lu & Mantei (1991), Bly & Minneman (1990) & Tang (1991) who have studied users working on a shared drawing surface while in separate locations.

7 Results of the Studies

Figure 4. Preparation for an essay by a Religious Studies major. Seven colours were used.

The system is now usable by individuals and small groups at a single workstation working on small scale projects. The users do not need to be computing experts. Figure 4 illustrates a design by an undergraduate majoring in Religious Studies who used the DNP to plan an essay about Witch Doctors. The various results are reported in detail elsewhere (Twidale et al. 1993a&b). Of particular interest for this paper is the degree to which the interface needed to be modified in response to the observations of use and the emerging requirements of users. This is despite the fact that we consider ourselves as having some experience in interface design. We believe that predicting what users will find difficult is an inherent problem in the development of any system of CMC which has many potential uses and users (Grudin 1988). For such systems it is extremely difficult to predict what users will do and hence to design an interface to best support that activity. We have observed a variety of styles of use of the system. For example, some people make great use of colours, others hardly at all. This variety may be due to three reasons:

1) Users have different styles of use. Both the process and product of designing vary greatly. This may be due to learning styles (Pask 1976), personal preference, academic background, etc.
2) We have observed the same user adopting different styles. This may be due to their working through an activity cycle such as brainstorm, rest, tidy up, reflect, brainstorm.
3) The nature of the task may determine the employment of a particular style.

Therefore the system and in particular its interface should be designed to accommodate a variety of styles of use.

The effect of the interface

A poor interface can have a substantial effect on performance, completely swamping the effect of

other features of the system. This can make evaluation of a system particularly difficult (Twidale 1993). An improved system with far greater functionality may fail to display the expected gains in terms of say greater student learning due to a poorly designed interface which requires the student to spend more time on learning and concentrating on how to operate the system and consequently less time on the subject domain.

When the interface for a feature is improved one can observe a dramatic increase in its usage. We found such a case for the Textnote feature. This usage change can in turn lead to a change in requirements. So with Textnotes easier to manipulate, some users created many more of them and then wanted additional features to search and manipulate them.

The co-evolution of system functionality and user requirements

It seems that users' requirements co-evolve with the developing system: as one feature is improved it changes the nature of the activity being supported and this in turn leads to a new requirement. This is analogous to the way word-processing features have developed. Word processors were developed to support the writing process, but when a writer is provided with an easy to use word-processor, its effectiveness changes the nature of the writing process (Norman 1988). This in turn leads to new requirements for word processors which when implemented will in turn affect the writing process, and so on.

Therefore user requirements and interface issues of usability are not just difficult to predict, they are inherently unpredictable and so continuous testing is a necessary approach to developing a successful interface.

Coping with a never-ending list of requirements

Partly as a result of this co-evolution and because of the ease of requirements capture by prototype criticism it seems that user requirements can become near infinite. Once you have a core system that is easy to use so that users can get started early on their designs they seem to articulate a never-ending stream of requests!

In particular it is important to discriminate between general and particular suggestions. Perhaps surprisingly, we believe it is more productive to heed the particular, especially at the early stages of design. A general suggestion has the characteristic form of "Wouldn't it be good if ...." whereas a specific suggestion is more like "What I want to do now is somehow to ....". The specific suggestion is couched in terms of the user's particular activity and needs to be translated into some feature that would allow the user to achieve that desired action (and to a form that is useful to others). General suggestions are particularly common from people observing a demonstration of the system but not using it to solve a problem of their own. A particularly recurrent suggestion is "Wouldn't it be good if you could do rough free-hand sketches?" Now this is a perfectly sensible suggestion and it is clear that this would be a valuable additional feature. However *none* of the volunteers using the system to date requested it, although they did ask for a host of other particular features.

It is our contention that this difference implies that it is better to improve the system's existing functionality of text plus simple annotations (so that users do not abandon using the tool for handling problems that are amenable to that approach) before extending the coverage to those problem types which require quite new functions such as sketching. Therefore for example the DNP would not be of much use to a mechanical engineer doing design because sketching is a crucial part of her design activity. However it is of use to many other kinds of user (computer scientists, arts/humanities undergraduates, project planners etc) who may occasionally employ sketches but for whom this is not a core activity. The reason it does appear to be useful to such people is that we have considered and implemented many of their specific suggestions that are generalisable and so useful to many kinds of user, but are incremental improvements of existing functionality rather than being completely new features. For example the framing option described in the system overview that allows you to group entities, move them and collapse them down to form a new subdesign (thus providing an abstraction mechanism) arose from a particular suggestion from a user who had a cluttered design after a bout of additional brainstorming.

It should be noted that implementing the particular suggestions is not a trivial panacea. As mentioned there are far too many of them. Therefore implementation needs to be prioritised according to several (potentially conflicting) criteria. These include the degree to which the suggestion might be generally useful (and the designer will need to consider how to make it so), its importance in impeding or accelerating a user's progress, its ease of implementation and the effect it may have on overall system performance, reliability and ease of use.

Although it might sound like mere common sense to focus on what people complain about before

adding in more features that they might want, there is a strong pressure to concentrate on the latter. Systems are more frequently compared by functionality (the number and variety of features, flexibility, generality etc) both in the commercial and the academic worlds. In the commercial world, after consideration of price, purchasers of systems are inclined to opt for the system with the most functionality and pay less attention to usability. This is understandable as 'usability' is hard to assess; functionality can be measured by looking at the list on the back of the box in many cases. Likewise in the academic world, a paper about a CMC that has more features, has a brand new feature, is more generic etc. is likely to appear more impressive than a paper that describes a 'mere' reimplementation, but this time attempting to make the system usable outside the lab it was developed in. The latter may be dismissed as development rather than 'proper' research.

Collaborative dialogues using DNP

A study of the educational potential of DNP was undertaken (Twidale et al. 1993b) using four groups of four computing undergraduate volunteers. These groups chose to use DNP to help them work on a piece of coursework which required them to work collaboratively on the specification of a deliberately open-ended and ambiguous remit. The intention of the coursework was for them to experience some of the creative aspects and problems of real world design. Of particular interest here is how groups of four users managed to successfully use a single workstation. The screen and the evolving design on it became the focus of interest and discussion. Although only one person could use the keyboard at a time and frequently one person used it for most of a session, all group members contributed to the problem and ideas once articulated on the screen seemed to become common property. That is they appeared not to be identified with the originator because all members participated in moving, shaping, colouring, positing or even renaming it by suggestions or consent.

Users' comments on their experience of using the tool often referred to its superiority over the confrontational nature of sitting round a table facing each other and writing on a piece of paper. When sitting 'shoulder to shoulder' round a screen, it was easy for any group member to take a turn in the conversation by leaning over and pointing at the screen. Even users who were sitting almost behind others would interrupt by leaning over and pointing. Because elements were easy to revise (rather than having to completely redraw them, as on paper), comments could be positive about how to change what existed. This differs from the pen and paper situation, which led to more negative comments in order to justify the effort of completely rewriting a design, or an attitude of deciding to put up with the current version rather than bother to change it. The ease of entry meant that the users claimed that they produced more and better ideas than working face to face.

8 Implications for CMC

Implications of the results

Although the DNP provides a new way of working, users were able to very rapidly learn to use it and organise their work to make productive use of it. This bodes well for novel ways of cooperative design and learning. However the very success of the shared screen in becoming a focus for cooperative dialogue and activity poses a challenge. For remote cooperation, it will be far harder to achieve this sense of a shared focus. Much research remains to be done on this.

The importance of turn taking by leaning and pointing at the shared screen implies that attempts to support synchronous distributed cooperation need to pay particular attention to the turn taking mechanisms employed. The lean and point method conveys at least two meanings: 'I want to talk now' and 'I want to talk about THAT'. Note that in some ways this turn-taking is more fair than in face-to-face discussion round a table where it involves split-second timing at the moment a speaker finishes. This can be particularly difficult for participants not using their native language.

Users also commented on the ease and speed of entering features and the lack of commitment with comments like 'Just type it in, we can always change it later'. If the system feels sluggish to the user it completely changes the nature of the interaction, particularly during periods of rapid revision activity. Users become aware of the system rather than their problem and are inclined to revert to using pen and paper. This effect will also need to be considered by developers of synchronous remote systems where computational delays are likely. Note that this reveals a potential danger of prototyping: a cut-down prototype may have an acceptably rapid response that the full version fails to achieve and so the final version may be less acceptable than the prototype.

The way that ease of revision enabled ideas to mutate and all participants to be involved in the activity meant that ideas were not usually associated with individuals but with the group as a whole. This has many advantages for criticising and further adapting the idea as well as supporting general group dynamics. There is a danger that the named contributions to conferences lack this advantage.

Conference organisers may wish to consider how to support a feeling of common ownership of an evolving idea within existing conferencing contexts. Whereas DNP only shows the final emergent design but not the process that led to it (unless partial designs are saved under different names), the text-conference is itself the change record, but lacks a summary of the final consensus (unless someone chooses to write this up and contribute it). It is likely that both the change record and a representation of the most recent idea structure would be useful.

Potential of the DNP in Computer Mediated Communication

The DNP offers users a means of rapidly articulating semi-formed ideas. It was intended to support the early stages of design and so was developed to allow software engineers to express vague creative initial ideas about a piece of software. Early designs are necessarily ambiguous and imprecise, but they are gradually refined to take on the structure and formalisms necessary to implement them in software. The same process can be used for many other creative activities including the exploration of ideas preparatory to writing about them.

Unlike pure text systems such as existing computer conferencing, the two dimensionality of the representations allows ideas to be expressed more succinctly. Links between words or their spatial connection can express subtle connections between them that have not yet been fully articulated (or even consciously considered yet). For example, two entities moved close to each other may mean; 'These are, or should be, somehow related, but I'm not sure how yet'. Later the connection can be considered and made more precise using labels on links, colour, shape, renaming the entities, creating intermediate entities, subdesigns etc.

The way that some conferencing systems use conferences and items to support different threads of conversation and the advantage this offers over a simple email distribution list is analogous to this, but in DNP the concept is taken further to within a single discussion. Indeed with the use of subdesigns, the system has features of a three dimensional textual representation, and a newly installed cross-referencing option (which creates a virtual link between entities in different designs) enables the construction of hypertext. Our results already show how this representation of ideas can be a powerful support for collaborative dialogues between groups of up to four people sitting round one screen. The screen becomes a focus of discussion and mutual misunderstandings can be detected and resolved by referring to and modifying the design. Yoder et al. (1989) have noted how a shared-database hypermedia system with an effective interface provides a powerful foundation for collaboration.

We next need to investigate the features that will need to be added to the system so that it can be effective for collaboration at a distance. A start has been made on this, using a distributed version of DNP which runs on two workstations. Using our gradualist approach, we placed the workstations next to each other and had the two volunteers sitting side by side, each using a separate machine. The idea was that this meant they had perfect audio and video bandwidth (they could look and talk to each other normally) and if necessary lean over and point to the other's screen. In fact, after deciding on how to proceed, they worked in silence. They read and attached Textnotes to an existing design structure they had collaboratively developed earlier. They worked by reading the other's attached Textnotes and writing and attaching responses on new Textnotes. Using a CMC viewpoint one could regard them as using the DNP as a computer conferencing system.

We next wish to investigate the use of DNP in asynchronous distributed collaboration. The designs can be converted into a textual form and sent as conventional email to be reconstituted at a remote machine. However it is likely that additional channels of communication would be needed to explain and debate a user's design or contribution. We can choose a simple case first of using the textnote facility to enable discussion of changes made and the meanings of designs, in a similar manner to that just described.

The next stage would be to accompany the DNP with a conventional textual computer conference. It could be eventually integrated into the conferencing tool so providing a means of communicating ideas in addition to conventional text. This would support asynchronous distributed collaboration. Once the interface issues for this scenario had been resolved to a satisfactory level, we could consider the synchronous distributed approach. This is likely to involve the integration of a shared design with video and audio links (Derycke & Vieville 1993).

Potential of the methodology

The employment of the DNP for remote synchronous CMC is a long term activity. However the results of our current research can offer a more immediate contribution to the design of CMC systems. Firstly the tool can be used as a planning tool for designing CMC courses. Course design is

a creative design activity involving the management of multiple constraints just like software design. A colleague in the Music Department at Lancaster is currently evaluating the use of DNP in designing hypertext modules for teaching elements of the undergraduate Music curriculum.

Also, our experiences lead to a set of recommendations for developers of CMC technology and course developers. The chief recommendation is to pay particular attention to the interface; systems developers should allocate a substantial proportion of resources to the evolutionary development and testing of the interface. The system developed by Alexander et al. (1993) is a good example of a careful approach, focussing on the development of appropriate interface features. Testing should begin at an early stage and involve regular and in-depth analysis of failures: features that users find hard to use. These tests need not be large scale; our experience is that it only requires a study of a few users to reveal glaring problems, but they should be frequent as the design evolves and interface errors are corrected. In order to begin testing as soon as possible, a minimal system should first be developed. Although this will naturally lack many features intended for the final system, it may still reveal significant problems which will be much easier to correct at such an early stage. (Some usage problems will be due to the absence of the full system functionality and so must be discounted, but our experience is that other, unexpected issues are revealed at a fortuitously early stage). The tests should be authentic; as close as possible in nature and style to the expected usage, even if of more limited functionality. This is important to reveal hidden assumptions in the design. An example of a suitable test for a distance education context would be to observe a volunteer attempting to dial in via a modem from their own home or office rather than in the ideal conditions of a development laboratory.

We have claimed that interface effects can swamp more subtle ones such as features of the system that can support collaborative dialogues. Therefore researchers wishing to investigate such features need to consider the interface with care lest any results they obtain are due more to an unexpected interface feature than to something more profound about the nature or problems of working with CMC. This swamping effect has already been observed in the domain of interactive learning environments (Twidale 1991) where a poorly worded prompt produced a whole new set of student errors that it would be easy to interpret as a substantial and widespread learner misconception about the subject domain.

Those introducing a CMC course who have the opportunity to choose the kind of conferencing software used should consider interface issues when making the software selection. It may be that a system with an easy to use interface will lead to better learning than one that has more sophisticated features but is less easy to learn and use. Whichever system is chosen, we would advocate increased proportion of resources to teaching users how to use the system. Ideas such as Carroll's Minimal Manual (Carroll 1990) would seem to be particularly relevant. Naturally if students do not learn how to use a CMC system effectively they are not going to learn the subject of the course effectively. We do not wish to repeat the early days of commercial word-processors where many such machines were abandoned in the corners of offices because secretaries received inadequate training in their use so that their manifest advantages over typewriters were not realised.

9 Conclusion

When developing and testing the DNP we are always aware that our system is in direct competition with pieces of paper (or maybe a whiteboard). The early stages of design are normally done on paper and designers are experienced at using paper. Our system is in competition with pen and paper in the same way that early commercial word processors were in competition with typewriters. If at any stage our volunteers (and ultimately users of the completed system) find the DNP awkward to use so that its longer-term benefits (of ease of revision etc.) are outweighed by short-term costs of clumsiness of use, they will abandon it and revert to using pen and paper. We have observed this in studies. For development purposes this is no problem: we just examine the situation when they gave up and try to improve the interface of the features they were then trying to use and match to their requirements. Outside user testing it is a problem. It does however impose a useful discipline on development in that we continually have to consider whether additional features might lead the user to give up and to go back to using paper or whether the feature postpones that moment.

In many current circumstances CMC systems do not have such direct competition. Users have to use the single system provided for the course. It may be that this lack of competition is why to date the evolution of interfaces for such systems has been slow. Instead, resources are devoted to adding features and enabling connection to many networks. These are both laudable aims, but they should not be pursued to the exclusion of interface features.

We can hope that the interest in interfaces will change in a manner analogous to that for Personal Computers. Users of DOS-based systems had to put up with the interface provided in the absence of

anything better, because many businesses required the purchase and use of PCs rather than say a Macintosh with its superior interface. With the introduction of Windows, the take-up has been extremely rapid and considerable debate now occurs on the development of better interfaces to applications running under windows.

Acknowledgements
Michael Twidale is a Science and Engineering Research Council Junior Research Fellow. The development of the Designers' Notepad was funded by the Joint Council Initiative in Cognitive Science and Human Computer Interaction.

References

Alexander, G., Lefrere, P., & Matheson, S. (1993). Towards collaborative learning at a distance. In F. Verdejo (Eds.), Collaborative dialogue technologies in distance learning (This volume).

Bly, S. A., & Minneman, S. L. (1990). Commune: A shared drawing surface. In Proceedings of the Conference on Office Information Systems, (pp. 184-192). Boston.

Carroll, J. M. (1990). The Nurnberg Funnel. Cambridge, MA: MIT Press.

Derycke, A. C., & Vieville, C. (1993). Real-time multimedia conferencing system and collaborative learning. In F. Verdejo (Eds.), Collaborative dialogue technologies in distance learning (This volume).

Ehn, P. (1989). Work-oriented design of computer artifacts. Hillsdale, NJ: Lawrence Erlbaum Associates.

Grudin, J. (1988). Why CSCW applications fail: problems in the design and evaluation of organizational interfaces. In Proceedings of the Conference on Computer Support Cooperative Work (CSCW '88), (pp. 85-93). Portland, Oregon: ACM Press.

Ishii, H., & Arita, K. (1991). ClearFace: Translucent multiuser interfce for TeamWorkstation (Research report No. NTT Human Interface Laboratories.

Lu, I., & Mantei, M. (1991). Idea management in a shared drawing tool. In R. M. Bannon L. Schmidt K. (Ed.), ECSCW'91, Amsterdam: Kluwer.

McGraw, K. L. & Harbison-Briggs, K. (1989). Knowledge Acquisition: Principles and Guidelines. Prentice-Hall, New Jersey.

Norman, D. A. (1988). The Psychology of Everyday Things. Basic Books.

Pask, G. (1976). Styles and strategies of learning. British Journal of Educational Psychology, 46, 128-148.

Shipman, F. M., & Marshall, C. C. (1993). Formality considered harmful: experiences, emerging themes and directions. Proceedings, InterCHI '93.

Sommerville, I. (1992). Software Engineering. Fourth Edition. Addison Wesley, Wokingham.

Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., & Twidale, M. B. (1993). Integrating ethnography into the requirements engineering process. In 1st International Conference on Requirements Engineering, San Diego: IEEE Press.

Tang, J. C. (1991). Findings from observational studies of collaborative work. International Journal of Man Machine Studies, 34(2), 143-160.

Twidale, M. B. (1991). Student activity in an Intelligent Learning Environment. Intelligent Tutoring Media, 2(3/4), 113-127.

Twidale, M. B. (1993). Redressing the balance: the advantages of informal evaluation techniques for Intelligent Learning Environments. Journal of Artificial Intelligence and Education.

Twidale, M. B., Rodden, T., & Sommerville, I. (1993a). The Designers' Notepad: Supporting and understanding cooperative design. In C. Simone (Ed.), ECSCW93, Milan.

Twidale, M. B., Rodden, T., & Sommerville, I. (1993b). The use of a computational tool to support the refinement of ideas. Submitted to Computers and Education.

Yoder, E., Akscyn, R., & McCracken, D. (1989). Collaboration in KMS, A Shared Hypermedia System. In Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems (pp. 37-42).