A MARKING BASED INTERFACE FOR COLLABORATIVE WRITING

Gary Hardock, Gordon Kurtenbach and William Buxton

Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 1A4

ABSTRACT

We describe a system to support a particular model of document creation. In this model, the document flows from the primary author to one or more collaborators. They annotate it, then return it to the author who makes the final changes. Annotations are made using conventional marks, typically using a stylus. The intent is to match the flow and mark-up of paper documents observed in the everyday world, . The system is very much modeled on Wang *FreeStyle* (Perkins, Blatt, Workman and Ehrlich, 1989; Francik and Akagi, 1989; & Levine and Ehrlich, in press). Our contribution is to incorporate mark recognition into the system and to explore some novel navigation tools that are enabled by the higher-level data structures that we use. The system is described and the results of initial user-testing are reported.

INTRODUCTION

Collaborative writing occurs in many different forms (Posner, 1991). Our system supports a model of document creation in which the document flows from the primary author to one or more collaborators. They annotate it, then return it to the author who makes the final changes. This flow of the document is illustrated in Figure 1. Annotations are created by marking-up a copy of the document with a stylus. The intent is to match the flow and the way paper documents are marked-up in the everyday world.

Other systems, such as Wang *FreeStyle*, emulate this model of document creation (Perkins, Blatt, Workman and Ehrlich, 1989; Francik and Akagi, 1989; & Levine and Ehrlich, in press). With *FreeStyle*, annotations are made not only with a stylus but also by recording speech. In our system annotations can only be made with the stylus.

With *FreeStyle*, the copy of the document that is distributed is only a "dumb" snap-shot of the original. Annotations are integrated as a separate layer to the "snap-shot." Furthermore, there is no computer recognition of the markings. To edit the final document, the user needs two

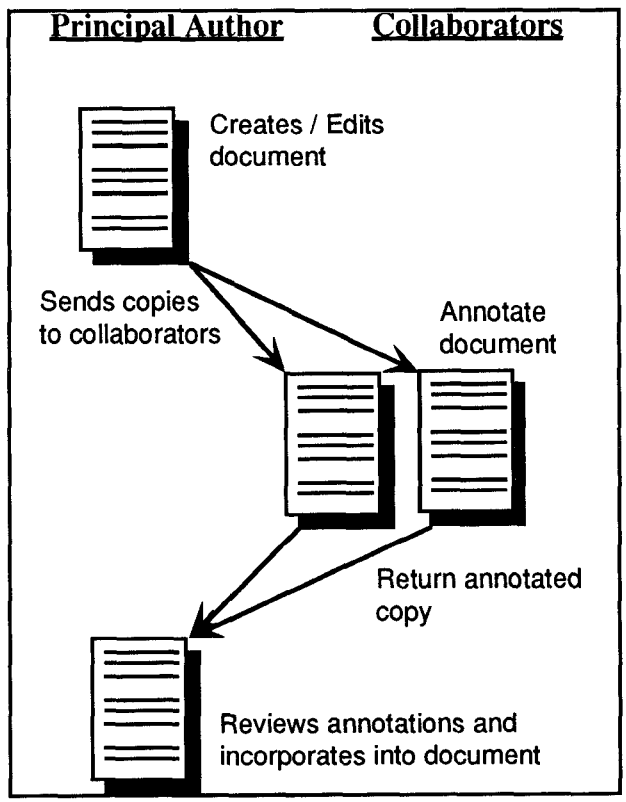

Figure 1: Asynchronous Collaborative Writing Scenario

The principal author creates a document and then sends copies of it to collaborators. The collaborators then annotate their copies and send them back to the principal author. The principal author then reviews the suggestions and possibly incorporates them in the newer version of the document.

versions: the "dumb" annotated copy and the original.

The system we developed is more sophisticated. The intention is to make the task of incorporating and managing changes to the document easier. Our system uses mark recognition to support automatically incorporating changes into the document, and provides novel techniques to help an editor avoid confusion when managing and navigating through a changed document. Mark recognition, change management, and navigation are made possible by using a data structure that is more sophisticated than a snap-shot of the document. Our document data structure contains information about the relationships between the annotations and text of a document. In what follows we describe the system and the results of our initial user-testing.

Goodisman and Goldberg have also built a system that allows documents to be edited and marked-up with a stylus (Goldberg & Goodisman, 1991; Goodisman, 1991). Their approach is based more on enhancing the actual manipulation of the text, whereas our approach is more concerned with a particular collaborative writing process. Both approaches should be seen as complementary as each addresses different issues of a much larger problem.

MATE: MARKUP ANNOTATOR TEXT / EDITOR

MATE functions in three modes, one for each step of our model of the writing process – "edit mode", "annotation mode", and "incorporation mode". The creation of the document can be achieved with MATE in edit mode or another word processor. In edit mode MATE functions as a marking-based text editor. (As a prototype, however, it is limited in functionality. It only supports *delete*, *move* and *insert* commands, as well as some novel *navigation* functions. Text is entered by the keyboard. Our concern is mainly with the other aspects of the processing of a document, as others have worked specifically on stylus-based text editors.) Figure 2 shows an example of MATE in "edit mode".

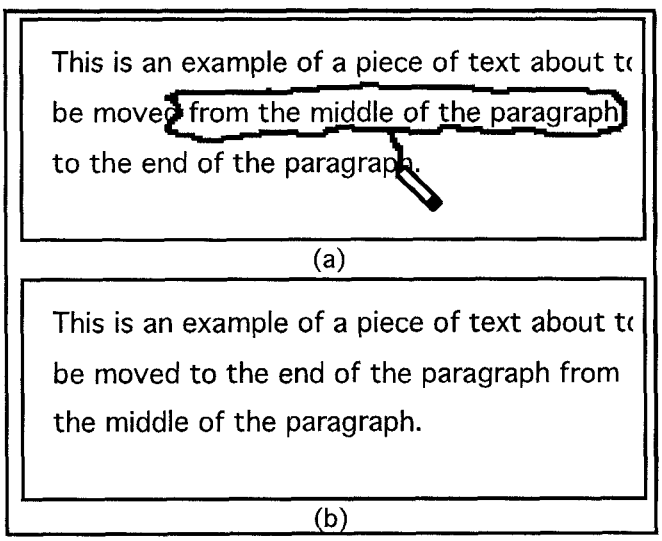

Figure 2: MATE in "Edit Mode".
(a) A user draws a move command. (b) After lifting the pen, the editing command is performed.

Copies of this document can then be e-mailed to the collaborators. Each reviews their copy using MATE in "annotation mode". In this mode a collaborator can specify suggested changes by marking up the document. As with paper documents and *FreeStyle*, markings are not interpreted as commands. They are treated strictly as annotations at this point in the writing process. The marks do not necessarily have to be editing commands. For example, a marking could be something as vague as "reword this paragraph". An example of various annotations is illustrated in Figure 3.

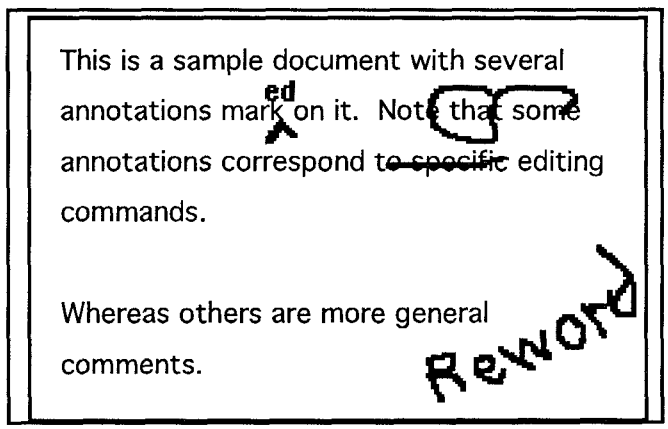

Figure 3: MATE in "Annotation Mode".
In annotation mode, users mark up a document in MATE just as if they were marking up a paper document.

Marked up documents are returned to the primary author, who does the actual revisions. This is done using MATE in "incorporation mode". In this mode, MATE displays two views of the document (Figure 4). The left view shows the marked up document received from the collaborators, with each reviewer's annotations appearing in a different color. Additional marks can be made, but the underlying text does not change, similar to annotation mode. The right view shows the current version of the document. No annotations are visible in this view as any marks made in this window are immediately interpreted as commands and executed immediately, similar to edit mode.

These two views work in concert. A user can point to an annotation in the left view and ask the system to perform it. The resulting changed document appears in the right view. The important characteristics of this design are that:

- the editing, annotation, viewing and incorporation tasks are integrated in a consistent, seamless manner.
- the annotations are visually persistent, even after they have been "executed", thus providing a mechanism to identify and select them at any time.

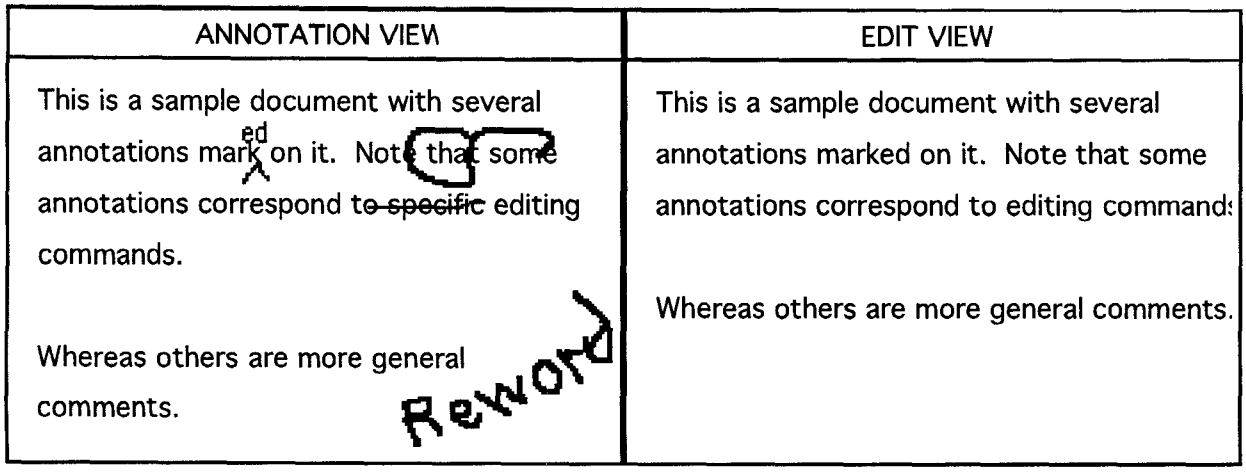

Figure 4: MATE in "Incorporation Mode".
In incorporation mode, a user can view the annotated document, and select which annotations to incorporate. Annotations t(seen in the left window) that have been "executed" appear as thin lines (e.g.., "ed"). Annotations that have not been executed appear as thick lines. Annotations are colour coded according to who made them. Annotations that represent commands can be executed by selecting them with the stylus. Annotations that have been executed can be "undone" by selecting them. The current state of the document appears in the right hand window. The user can navigate (scroll) independently or synchronously in each window.

Other reasons for this split view design are identified later when the specific design issues are discussed.

COMBINING ANNOTATIONS AND EDIT COMMANDS

Marks have already been used extensively in annotating documents (Chow and Kim 1989; Carr 1991, Welbourn and Whitrow 1988; Wolf, Rhyne and Ellozy 1989). They are also being used to specify commands to computer applications (Carr 1991). There are several properties of marks which make them good for each purpose:

- marks are visible; they provide a high contrast between figure – markings, and ground – text.
- people spend many years learning how to make and understand marks
- marks allow a very flexible protocol
- marks are spatially laid out.

All of the above properties make marks well suited for annotating documents. The fact that most people can make and understand marks gives a compelling reason to build mark-based interfaces to computer applications. But it is because marks possess all of these properties that enables the same marks to be used as both an annotation and as an editing command.

In short, efforts in marking interfaces have been mainly directed at human-computer interaction. Perhaps even more important, from our perspective is their value in computer-mediated human-human communication. (Hence the effectiveness of FreeStyle, despite the absence of any mark recognition.)

One way of comparing the annotation process and editing a document on a computer is that in the first case a person is communicating with another person, whereas in the second case a person is communicating with a computer. The goal then becomes to design a method in which a person can communicate to both another person and to a computer application.

The fact that marking commands to the computer application are visible is the key. If, instead of immediately interpreting a marking command, the computer does not process the mark but simply leaves the mark visible, the mark can be thought of as an annotation. From this point of view, annotations are deferred edit commands. In terms of the different modes of MATE, annotation mode can also be called *deferred mode*, and edit mode is *immediate mode*.

FUNCTIONALITY AND CAPABILITIES OF MATE

In addition to the benefits and issues concerning the use of markings for editing and annotating, there are many advantages and issues when both uses are combined in an integrated system. In this section we describe some of these.

Do by Selection:

As mentioned earlier we can select markings in the annotation view to be performed in the edit view. The selection is accomplished by tapping on the marking. Feedback is provided by making the marking thinner, indicating that it has been performed. Note that this selection is possible only when both the user and the computer can understand the editing command identified

by the marking. Figure 4 shows the result of selecting the insert and delete annotation marks for incorporation.

History Mechanism

The feedback provided by thinning the markings which have been performed provides a history of the annotations which have been incorporated. However, this graphical history mechanism is spatial in nature, not chronological. This is much more useful as the order of incorporating annotations is unimportant, whereas the locations of the annotations immediately tell us what text the commands have been applied to. For example, in Figure 4, we immediately notice that the insert and delete annotations have been performed and to what pieces of text they were applied to, but we do not know when or in what order.

Undo by Selection

Just as a mark can be incorporated by tapping on it, a mark can also be unincorporated by tapping on it again. This is only possible because the annotation marks are always visible, a result of having the two views of the document. Note that Undo by selection is not order dependent. Figure 5 shows an example of how this works. It does not matter whether the delete or move command was performed first, either command can be directly undone. Note that the annotations correspond to a specific piece of text, not the position of the annotation. Such a mechanism for undoing text-editing commands has never been built before.

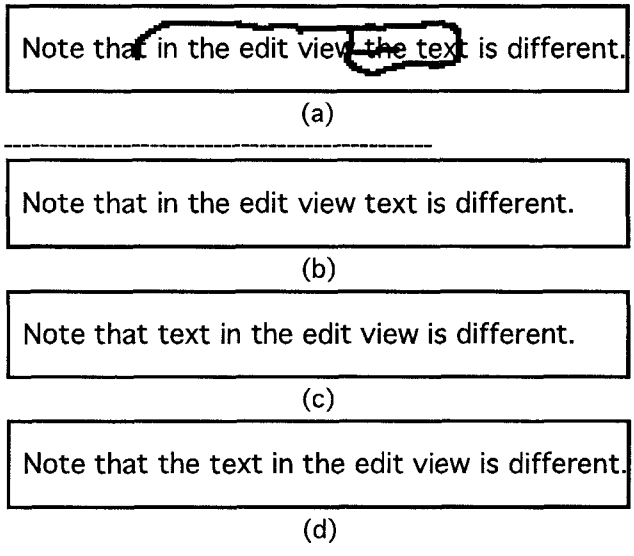

Figure 5: Undoing irrespective of temporal order.
The initial text is shown in (a). (b) is the result after the deletion is performed. (c) is the result after the move is performed. After undoing the delete command, the desired result is (d).

Previewing

By combining Do and Undo by selection we have a method for previewing the results of suggested annotations. If a user wishes to preview what would happen if they incorporated an annotation, they can Do it, examine the results, and then Undo it. Note that because the Undo is non-chronological any combination of annotations can be previewed. This allows the user to try "what if" scenarios, with only the cost of tapping on the annotations. Note that the user need not be the principal author. It could be used by a collaborator wanting to see the results if the annotations were incorporated.

Enhancement of Understanding

The ability of the system to understand the meaning of a mark lessens the possibility of a misunderstanding between the annotation's writer and reader.

- everyday use brings certain expectations, namely the maker of the annotation expects the person interpreting the annotation to understand the markings.
- The reader may not meet the writer's expectations in terms of understanding the writer's annotations.
- But the computer can give immediate feedback to the writer to verify the writer's expectations about the computer's understanding.

Table 1 shows that the addition of computer recognizing commands can aid in the communication between the annotation's creator and reader.

Viewing the Annotations of Several Authors

The annotations of several collaborators can be overlaid in the annotation view. To differentiate among the various collaborators markings, each set is displayed in a different color. Note that this is possible because the markings are separate from the text document, so different characteristics can be added to each set of markings. This is analogous to writing on a transparency overlaid on top of a paper document and writing on the transparency. Then several marked up transparencies can be placed over the paper document.

A preliminary study using transparencies was conducted to identify the issues and advantages of overlaying the annotations. The results indicated that the usefulness of this feature depends upon the density of annotations on the page. In some cases a single transparency was cluttered, and overlaying several transparencies made the annotations illegible.

As the problems of clutter are also found in a single set of annotations, we decided to address the more general problem of reducing the density of the markings. This is accomplished by providing support functions such as "Hide Set of Marks", "Show Set of Marks", "Hide Mark", and "Show Hidden Marks".

Broken Move, Multiple Buffers and Placeholders

As the editing marks also serve as annotations, we have examined the markings used in the everyday marking-up of

Understanding	Computer		
Reader	Understands	MisInterprets	Does not Understand
Understands	mutual understanding	user will ignore computer's misinterpretation	computer neither aids nor hinders user interpretation
MisInterprets	Computer understands, may help the reader understand the marking	user and computer may have different misinterpretations, which may cause the user to possibly try a different interpretation	
Does not Understand		User may accept the computer's misinterpretation	

Table 1: Enhancing the Understanding between the creator and reader of an annotation.
If the reader does not understand or misinterprets the meaning of a marking, the computer may be able to understand it

paper documents to gain insights into the design of the editing commands. One command we have "borrowed" from pen and paper is the *broken move* command, shown in Figure 6. Instead using a "star' symbol to move to and from, we use a marking menu containing various symbols termed *placeholders*.

Another way of thinking about this command, besides as a broken move, is as two commands, move to buffer with ID, and move from buffer with ID. This allows multiple text buffers, each with a unique placeholder symbol to identify it. It is important to note that the user chooses which placeholder to move text into, thus providing a strong connection between the placeholder and the text it contains. Also, because the placeholders containing text are visible, it is trivial to determine what text is contained in a placeholder buffer.

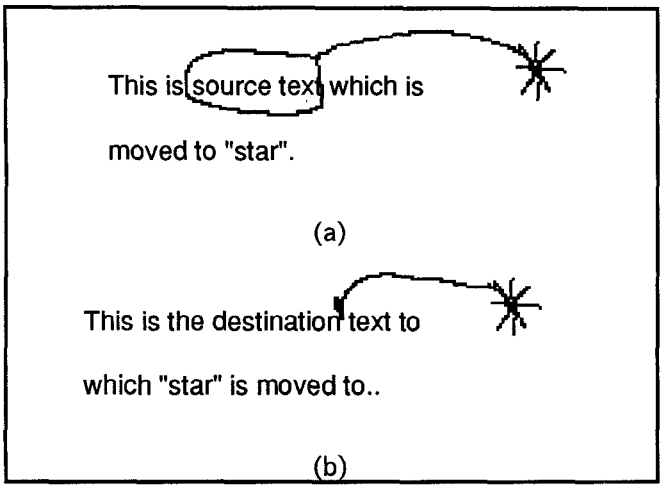

Figure 6: Moving text across pages using "move to star" and "move from star".

THE SUPPORTING USER INTERFACE

The overall design of the user interface allowed the system to provide the functionality mentioned above, but there are several additional interface components which are needed to support this functionality

Interacting with the Markings

We mentioned above that markings could be done and undone by tapping on them. But other operations on the marks are also necessary, for example, the need to hide or erase marks. In order to provide this functionality marking menus (Kurtenbach & Buxton, 1991) were implemented. This gives a logical extension to tapping on the mark. If the user taps on the mark, it is done or undone. If, however, the user makes a mark starting from an existing mark it is interpreted as a marking menu selection. Figure 7 shows the marking menu for operations on marks. GoTo and GoBack are discussed in the section on coordinating the two views.

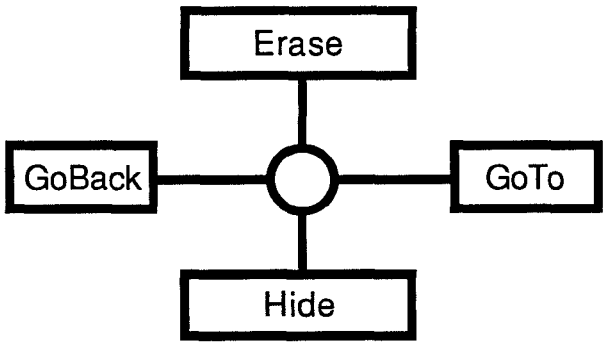

Figure 7: Marking menu for operations on annotation markings.

Navigation

Instead of using scrollbars, our approach to navigation was based upon the handling of a piece of paper directly with the pen. By holding down the middle button of the mouse

or the button on the pen, the user enters *navigation mode*. In this mode, the "page" or window of the annotation or edit view can be grabbed and moved. An upwards motion moves the document forward, which corresponds to moving the scrollbar or viewing window down. If the pen motion is slow the document moves as if it has been grabbed and is being pushed. If the motion is faster then the document moves as if the user had *flicked* a page forward or backward.

Independent Direct Navigation. Navigation mode is implemented as a modified marking menu, shown in Figure 8. Page flicks are always too fast for the menu to pop up, but the page push commands can be selected by either the mark or the menu. These "flicks" and "pushes" are applied to the annotation and edit windows independent of each other.

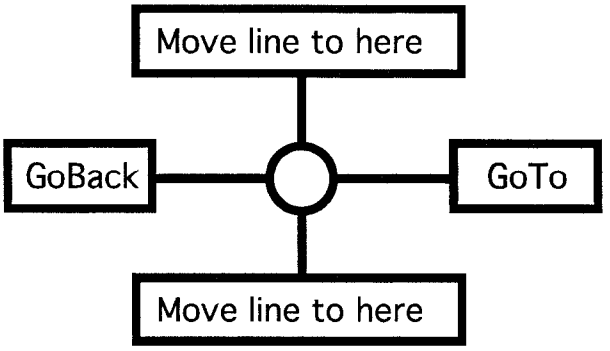

Figure 8: Marking menu for navigation.

Coordinating the Annotation and Edit Views. Several problems appear due to the fact that there are two views of the same document. Some of these are:

- how should scrolling and other types of navigation be coordinated between the two views?
- what should happen, and what should appear after an editing command is performed, or an annotation is selected to be done or undone.?
- what support is needed to aid the user in understanding the relationship between the two views?

Linked Direct Navigation. As the document changes, the annotation and edit views become more and more different. Therefore it is not always clear how each view should react when a navigation command is performed in the other view. One solution is to "link" the two views so that they move equal amounts. This is accomplished by modifying the page flicks and page pushes. If the user wants a linked movement, he or she first makes appropriate move command and then without lifting the pen, draws a line to the left or right. Note that in some cases the line will look like an "L", thereby providing a mnemonic cue.

Context Dependent Linked Navigation

Whereas, the direct navigation mechanisms are based upon relative movements, the context dependent navigation allows the user to align the two views based upon a specific piece of text – context. This is accomplished via the GoTo and GoBack commands.

There are two slightly different types of GoTo commands. When GoTo Text is applied to a piece of text in one of the two views, the corresponding piece of text is found in the other view, and then the other view is aligned with the first view. The piece of text is also highlighted.

GoTo Annotation is similar to GoTo Text, except that it is applied only to annotations. This means that GoTo Annotation can only be used in the annotation view. When GoTo Annotation is applied to a mark, the mark is interpreted, and if the interpretation is successful, the Edit view is aligned with the annotation view and the text affected by the command is highlighted.

INTERFACE ALTERNATIVES

The general concepts discussed in the introductory sections could possibly be applied using interfaces other than those based upon the pen and paper metaphor, but the property that marks are visible and the capability to use the same mark as both an annotation and for specifying the editing command gives the marking-based approach definite advantages.

GUI Interfaces

For example, a GUI type of interface with a mouse and keyboard could be used to enter comments and specify commands. This was partially achieved with the Collaborative Annotator (Koserak et. al 199?). However it only allows annotations to be made, it is not a text editor. The main problem with GUI type or direct-manipulation interfaces is that there is a separation between specifying editing commands and specifying annotations. The actions used in specifying editing commands, such as menu selections, mouse movements, and button clicks do not leave a "visible audit trail" (Kurtenbach 19??), and therefore cannot be easily used as annotations.

Speech Interfaces

Speech is a very good means of communication among people, therefore it makes sense to use speech as a method of annotating a document. But speech is not visible and is poor in specifying locations. Also, speech is good for certain types of annotations, but poor for others. In fact, speech is best for general comments and "wordy" comments, exactly the types of annotations for which mark-up annotations are poor for. Therefore, like *FreeStyle*, the ideal system would use a combination mark-based speech based interface. However, as the speech based-interface is dependent upon the mark-based interface, we decided to first see would could be gained

with a purely mark-based approach. A hybrid system is left as future work.

Alternatives to the Two Views

Several problems occurred as a result of having two views of the document, the annotation and edit views. There are alternatives, each of which has its own advantages and disadvantages.

One View for Each Set of Annotations. This is a good solution for the multiple sets of annotations problem. But it magnifies the problems in coordinating a single annotation view with a single edit view. Although the benefits of multiple annotation views may outweigh the disadvantages caused by these problems, we decided to concentrate on the issues concerning the coordination of one annotation and one edit view first and leave multiple annotation views as future work.

Single View. Another alternative is to have a single view. Although this might seem to solve the coordination problem, it only transforms it into another related problem. As the document in the single view changes, the annotations will have to be modified to adapt to fit the current version of the document. This is possible for annotations which can be interpreted as editing commands, but for general comments there is no way for the system to know how they should be transformed. this is illustrated by the example shown in Figure 9.

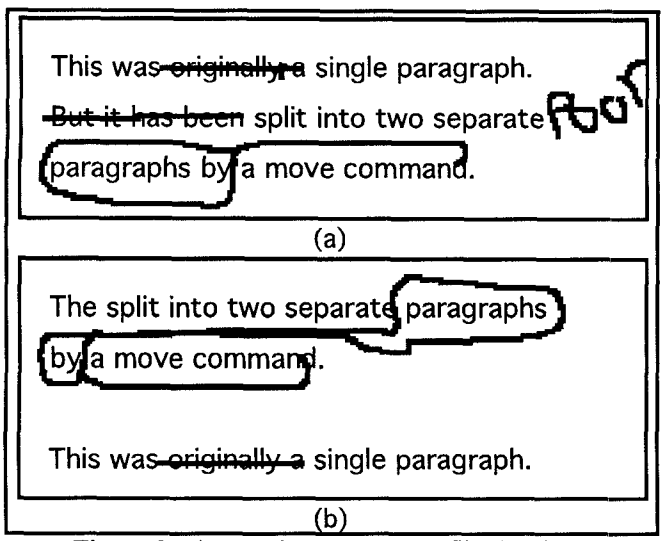

Figure 9: Attempting to Update a Single View.
(a) shows the original view, before the update. (b) shows the result of an attempted update. Such an update is very difficult as almost all of the marks need to be changed or moved, including marks strictly meant to be treated as comments. Note that the placement of "Poor" is unknown.

These arguments do not mean that the two view approach is necessarily better. It is an alternative with its own issues and problems. It would be useful in the future to design the alternative interfaces and compare the advantages and disadvantages to each approach.

EXPERIENCES WITH MATE

MATE is still at the preliminary design / implementation stage but some limited user testing has been done. The main purpose of the user testing was to determine if the underlying concepts were valid, and to identify any major problems with the system. Also as the navigation command set was fully functional, it was tested in more detail.

Five users were placed in a mock scenario, in which they were to pretend that they were the principal author given a marked-up document to edit. What they chose to incorporate was entirely up to them.

The results of the study showed that users choose to incorporate annotations by selection rather than by manually doing the edit themselves. Issues concerning histories, undo, and previewing were inconclusive as the users did not use these features.

The task was not complicated enough to bring out the issues regarding the coordination of the two views. However, there was some confusion between the GoTo Text and the GoTo Annotation commands.

The navigation testing was confounded by the stylus which was unreliable. In particular this made page flicks very difficult to specify. Several users preferred to use the scrollbar, instead of page flicks. Another problem was the response time of the system for page movements, which often confused users about whether it understood their command or not. These results do not conclude that page flicks do not work, but that more fine-tuning will be needed if they are to be useful and usable.

SUMMARY AND CONCLUSIONS

MATE was designed and built as one solution to the asynchronous collaborative writing problem. As mentioned above, there are several alternative approaches and many directions for future work. MATE is intended to be a first step in identifying, and addressing some of the problems with asynchronous collaborative writing. Also several new issues and uses of marking-based interfaces have been developed in the process.

ACKNOWLEDGMENTS

We would like to thank the members of the Input Research Group at the University of Toronto who provided the forum within which this work was undertaken. Primary support has come from Digital Equipment Corp., Xerox PARC, and the Natural Sciences and Engineering Research Council of Canada. Additional support has been provided by Apple Computer Inc., IBM Canada's Laboratory Centre for Advanced Studies, and the Arnott Design Group of Toronto. This support is gratefully acknowledged.

REFERENCES

Chow, D & Kim, J. (1989). Paper-Like Interface for Educational Applications, *National Educational Computing Conference '89*, Boston, Massachusetts, pp. 337 - 344.

Francik, E. & Akagi, K. (1989). Designing a computer pencil and tablet for handwriting. *Proceedings of the Human Factors Society 33rd Annual Meeting*, 445-449.

Carr, R. M. (1991). The point of the pen. *Byte*, February, pp. 211-226.

Goldberg, D. & Goodisman, A. (1991). Stylus user interfaces for manipulating text. Proceedings of the Fourth ACM SIGGRAPH Symposium on User Interface Technology (UIST'91), 127 - 135.

Goodisman, A. (1991). A stylus-based interface for text: Entry and Editing. Technical Report CSL-91-10, Xerox Palo Alto Research Center.

Hardock, G. (1991) Design Issues for Line Driven Text Editing/Annotation Systems. In *Graphics Interface '91*, Calgary, June, 1991. pp. 77 - 84

Kosarek, J. L., Lindstrom, T. L., Ensor, J. R., & Ahuja, S. R. (1990). A Multi-User Document Review Tool. In S. Gibbs, & A. A. Verrijn-Stuart (Ed.), *Proceedings of Multi-User Interfaces and Applications* (pp. 207 - 215). North-Holland: Elsevier Science Publishers.

Kurtenbach, G. & Buxton, W. (1991). Marking Menus In *UIST: Proceedings of the ACM Symposium on User Interface Software and Technology*. pp. 137 - 144.

Levine, S.R. & Ehrlich, S.F. (in press). The Freestyle system: a design perspective. In A. Klinger (Ed.). *Human-Machine Interactive Systems*. New York: Plenum Press.

Perkins, R., Blatt, L., Workman, D. & Ehrlich, S. (1989). Iterative tutorial design in the product development cycle. *Proceedings of the Human Factors Society 33rd Annual Meeting*, 268-272.

Posner, I.R. (1991). A Study of Collaborative Writing. Master's Thesis, University of Toronto.

Welbourn, L.K. & Whitrow, R.J. (1988). A Gesture Based Text Editor, in D. Jones & R. Winder (Eds.), People and Computers IV, Proceedings of the Fourth Conference of the British Computer Society Human-Computer Interaction Specialist Group. Cambridge, Cambridge University Press, pp. 363 - 371.

Wolf, C.G., Rhyne, J.R. & Ellozy, H.A. (1989). The Paper-Like Interface, IBM Technical Report RC 14615 (64399) 2/3/89, also in Designing on Using Human Computer Interfaces and Knowledge-Based Systems, G. Salvendy & M.J. Smith (Eds.), Elsevier Science Publ, Amsterdam, 1989, pp. 494 - 501.