Annoflow

Phil Crosby *Michael Quinn* *François Guimbretière*
Department of Computer Science
Human-Computer Interaction Lab
University of Maryland,
College Park, MD, 20742
philc@philisoft.com mikejquinn@gmail.com francois@cs.umd.edu

ABSTRACT
Proof-reading digital documents is a difficult task, because the marks made on documents do not maintain their relevance as the document changes. In addition, applying the changes indicated by the proof-reading marks to the document is tedious and error-prone. We propose Annoflow, a system for managing document annotations. Annoflow manages both free-form margin annotations and proof-reading marks, intelligently reflowing them as the document changes to maintain their relevance. It also interprets and applies the changes indicated by ANSI proof-reading marks made on the document. In this paper, we use Annoflow to evaluate through user study different methods for applying proof-reading marks, and measure user confidence in the system.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Reliability

Keywords: Margin annotation, proof-reading, reflow

INTRODUCTION
Annotating documents is an important function for proof-reading tasks. Existing systems like Microsoft Word [3] and XLibris [1] enable digital document annotation and integrate the annotations directly into the document. However, document authors often want the annotation information to stay meaningful and relevant as the document changes. Additionally, when authors review proof-reading marks made by themselves or others on their digital documents, they have to manually interpret all of the marks and make the changes to the document themselves. This can be a tedious and laborious process. Authors need a faster way to apply proof reading suggestions to their documents. To address these needs, we propose Annoflow, a complete implementation that manages annotations penned in Microsoft Word, preserves their meaningfulness as the document changes, and automatically applies standard ANSII proof-reading marks. Using this system, we compare strategies for managing annotations and evaluate the best ways to facilitate proof-reading on digital documents.

UIST'05, October 23–27, 2005, Seattle, Washington, USA.

Several solutions exist that address sub-problems in the area of managing document annotations. Grouping handwritten strokes into meaningful units is important for distinguishing between different freeform annotations. Dynomite [4] is a note-taking system that clusters strokes into groups based strictly on time; other systems [2] rely on temporality, proximity, context in the document, and the shape of the strokes. Our implementation uses a mix of two techniques for grouping: explicit cues from the user, and the Tablet PC SDK grouping implementation, which uses proximity and temporality criteria to form groups of strokes.

"Reflow" is positioning the annotations intelligently as the document structure changes. ProofRite [5] demonstrates accurate reflow of proof-reading marks, but does not address margin annotations. Microsoft Word supports annotation to documents, but does not reflow marks made in the body of the document. Also, Word supports margin annotations only on the right margins of a document, which can be anchored only to individual words, and cannot be "linked" to words or strokes made in the body of the document via call-out marks. Several tactics for reflow have been proposed in the Callisto system [6], but reflow for margin annotations is limited to moving them vertically up and down as their paragraphs move. Our system robustly reflows both document proof-reading marks and margin annotations, handling annotation collision and rendering of anchoring marks.

In addition to managing margin annotations, we have also developed a solution for interpreting and applying standard ANSI [7] proof-reading marks. There are few solutions in this area that apply to digital documents. ProofRite is able to recognize and anchor only a few single-stroke proof-reading marks, and it is left up to the user to manually apply them to the text. Our system can recognize a much greater subset of ANSI proof-reading marks, and can actually interpret them and apply the desired changes to the document.

MANAGING PROOF-READING ANNOTATIONS
Proof reading annotations are made in the body of the document. To stay relevant as the document changes, they

must anchor accurately, reflow sensibly, and be recognized and interpreted correctly.

Recognizing Proof-Reading Annotations

Our system leverages a slightly modified version of the Siger recognition engine [8] to design recognizers for proof-reading annotations. Strokes are first broken up into a series of vectors. Then, the direction of each vector is encoded into a string.

For each stroke in an annotation, we've designed regular expressions to describe these directional patterns. These regular expressions are then matched against the encoded string for the new strokes. For greater accuracy, we use many additional stroke characteristics, such as the percentage of the stroke traveling in a specific direction and the distance between the stroke's start and end points.

Clustering and Anchoring Proof-Reading Annotations

Some annotations consist of more than one stroke. To recognize multi-stroke proof-reading marks, we must cluster strokes that might be related, and then analyze them as a whole unit. Our system does this by keeping track of all "incomplete" annotations. When a new stroke is added, it is first determined whether or not the new stroke is recognized as a part of a multi-stroke annotation. If it is, the new stroke is checked against all known incomplete annotations. An incomplete annotation can then "claim" the new stroke as its own if it is determined that the strokes are associated with one another, which is determined by the new stroke's proximity and its shape.

After an annotation is interpreted, it must be anchored to a point in the document to allow for proper reflow. For each supported proof-reading mark, we have identified a specific point within its strokes that is used for identifying the text most relevant to the annotation. Annotations are anchored to the word or character in the document which lies directly below the stroke's anchor point.

The anchor points are annotation dependent (Figure 1). Many annotations, such as the "italic" proof-reading mark, simply use the midpoint of the stroke. One of the more interesting anchor points is that of the transpose mark, which uses the point of inflection between the concave-down and concave-up halves of the stroke (Figure 2). We find the point of inflection by associating a vector with each point of the stroke. The vector with its angle closest to 90 degrees (pointing down, if written left-to-right) is determined to be the point of inflection.

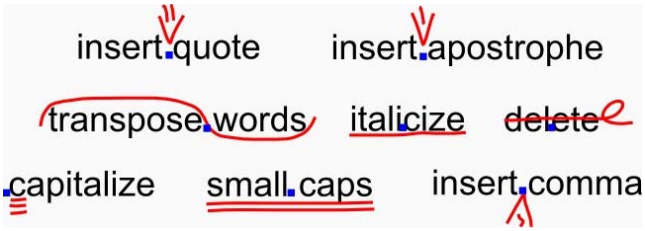

Figure 1: Common proof-reading marks and their anchor points. Anchor points are determined by heuristics specific to each mark.

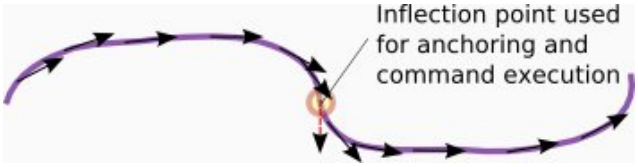

Figure 2: The transpose mark's anchoring point is determined by the inflection point of the curve.

Some annotations require contextual information from the document to be anchored accurately. For instance, the "insert period" and "insert comma" annotations are always meant to be placed directly behind a word. We look at the words surrounding the anchor point and use this information to fine tune the accuracy of where the annotation is anchored.

Strokes which are not recognized as ANSI proofreading marks are simply anchored to the page at the stroke's midpoint.

Reflowing Proof-Reading Annotations

Proof-reading annotations within the document must properly reflow as the document changes in order to remain relevant. Our system anchors proof-reading marks to words or characters, which then follow the text as it reflows.

The problem of reflowing annotations that extend across multiple words has been previously addressed by both XLibris and ProofRite. As such, our system only allows for proof-reading annotations that apply to a single word or character within a word, with the exception of the "transpose" proof-reading mark.

If a word is deleted that contains a proof-reading mark, the mark is deleted along with the word.

Applying Proof-Reading Annotations

Each proof-reading annotation contains an associated action which it performs during when the proof-reading mark is applied to the document. The annotations that only apply to a specific point in the document (such as the "insert period" annotation) execute directly at their anchor point. Other annotations may extend across an entire word (such as the "italic" annotation), and so we must use the context surrounding the anchor point to accurately apply the proof-reading mark.

When to apply the proof-reading marks is a difficult question. Our system affords two options: applying them as soon as they're recognized, or applying them all at once. Each of these is tested and discussed in our user evaluation.

MANAGING MARGIN ANNOTATIONS

We encourage users of our system to use an explicit interface for grouping and anchoring their margin annotations; when this interface is neglected, we fall back to an implicit grouping and anchoring system.

Explicit Marking Interface

Annotations made in the margins of documents can have grouping and anchoring ambiguity, even for humans [Callisto]. Users recognize these ambiguities and can become frustrated in trying to "guess" what the system will do. To solve this problem, we've designed an explicit marking interface that eliminates the ambiguity and guesswork.

After writing an annotation, the user can draw "ticks" around the corners of the annotation which explicitly groups the strokes together. Furthermore, to anchor the annotation to a paragraph, the user can draw a straight line from the annotation to the desired paragraph it's referring to. Margin annotations can also be linked to inline annotations by connecting the two with a line (Figure 3). This explicit marking interface eliminates ambiguity and error in grouping and anchoring, and raises the confidence in the system's accuracy.

Implicit Marking Interface

It's unreasonable to expect the user to perfectly adhere to the explicit marking interface all of the time. In fact, having to continually draw tick marks around every annotation can become burdensome, and is unnecessary in cases where it's fairly obvious how the annotation should be interpreted.

When no explicit grouping marks are present, we anchor and group according to some reasonable defaults. For grouping, we run the strokes through the Tablet PC recognizer, which gives fairly accurate groups according to words, sentences and paragraphs. Any strokes that are classified as being in the same paragraph are grouped together. Grouping doesn't discriminate based on temporality; if a mark is written near or on top of another annotation after the fact, it is grouped together with that existing annotation based on position alone.

In the absence of anchoring marks, the annotation is anchored to the paragraph located nearest to the top of the bounding box of the annotation.

Reflowing Margin Annotations

Like proof-reading marks, margin annotations must adapt to changes in the document to remain relevant. Annotations made in the margin are anchored to a paragraph either implicitly or explicitly, and then move vertically in the margin as that paragraph moves up and down in the document. If a paragraph is deleted, we interpret this as meaning that any annotations anchored to that paragraph are no longer relevant, and we remove those annotations from the document.

When annotations written in the margins are "linked" to a mark made in the body of the document ("inline mark"), the margin annotation must be sensitive to how the inline mark moves. In our design, when the inline mark moves around in the paragraph, the margin annotation stays fixed; only the anchoring mark is updated (Figure 3). We avoid "cleaning up" the anchor mark and redrawing it as a smooth line; instead, we favor preserving the look of the user's anchor mark.

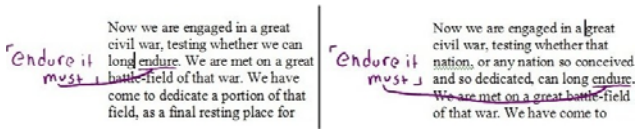

Figure 3: Anchor marks transformed to remain connected as the document annotation is moved.

When the inline mark changes paragraphs, the margin annotation is moved along with it. This is to avoid very long anchor marks stretching across the page.

As the document is changed, margin annotations may run into each other and overlap as they're reflowed around the document. To prevent this, we've designed an algorithm to optimize the position of annotations in the margin so that they avoid overlap and yet still remain meaningful. Briefly, our overall strategy is to make as little disturbance to the annotations as necessary. Our algorithm shifts margin comments up and down to make room for a newly reflowed annotation. It performs a multi-pass location optimization to fill all available white space. It reflows an annotation above or below the bounds of the paragraph it's anchored to only as a last resort, and never moves an annotation to the opposite margin.

IMPLEMENTATION

Our implementation is built as a Microsoft Word add-in, using the Tablet PC SDK and its recognizer. Some annotations are recognized using a modified version of the Siger recognizer, while others use the Tablet PC SDK's built-in gesture recognizers.

While working with the internals of Word, we've found that our task could have been made considerably easier if it supported the concept of anchors that can be attached to a word or paragraph, which inform listeners as they're moved in the document. Currently, we do this in a circuitous fashion by inserting a 1x1 pixel "shape" into the document, which holds a custom Windows Forms control that reports any changes made to its position.

Another useful addition to Word's SDK would be access to its drawing layer. Currently we attach an ink-collecting overlay to Word and use that as our drawing surface. Since Word doesn't report when it needs to redraw, it can periodically overwrite what's displayed on the overlay, causing flicker.

USER EVALUATION

(This study hasn't been conducted yet).

We're planning to conduct a user study that will try and discover what is useful for proof-reading tasks by evaluating our proof-reading annotation reflow and execution.

We will measure the performance of our system by asking users to manually classify the correctness of reflowed marks, and the correctness of their application to the document. We will then ask whether this level of accuracy is trustworthy enough to warrant use. We anticipate that anchoring, reflow, and execution accuracy will be greater

than 90%, and that this will be adequate for users to trust the system.

We're going to test which is more effective: instant-apply of proof-reading marks as they're written, or applying them later all at once. Instant apply offers immediate feedback and gratification. If a mistake is made, it can be corrected right away. However, having your document change as you write on it can be distracting, and applying annotations immediately introduces ambiguity in some multi-stroke commands.

Delaying the execution of proof-reading marks is more optimal in peer review scenarios, where only the author wants to make changes. This model decreases confidence, because you cannot be sure if the system interpreted the stroke correctly. To alleviate this problem, we've created an undo sidebar that lets the user undo the application of any proof-reading mark (**show in figure**).

Our hypothesis is that the advantages of delayed execution will overcome confidence problems because of our undo framework, and will be the preferred method of applying proof-reading commands.

DISCUSSION AND FUTURE WORK

(Discussion of results)

In future work, we hope to measure the accuracy of our system for margin-reflow tasks. We also wish to get user feedback on how well our explicit and implicit margin annotation interfaces work. It is less obvious how margin annotations should be anchored and reflowed, and there are many different reflow strategies. Using Annoflow, we wish to compare different reflow strategies and see which is most effective for document annotation.

ACKNOWLEDGMENTS

REFERENCES

1. Schilit, B.N., Golovchinksy, G., and Price, M.N. Beyond paper: supporting active reading with free form digital ink annotations, in *Proceedings of CHI98*, 249-256.
2. Golovchinsky, G. and Denoue, L. 2002. Moving markup: repositioning freeform annotations. In Proceedings of UIST '02, pp. 21-30.
3. Microsoft Word - missing
4. Wilcox, L. D., Schilit, B. N., and Sawhney, N. 1997. Dynomite: a dynamically organized ink and audio notebook. In Proceedings of CHI '97. pp 186-193.
5. Conroy, K. Levin, D. Guimbretière, F. 2004. ProofRite: A Paper-Augmented Word Processor. In Proceedings of UIST '04. [Is this the best version to cite? I can't find its citation online.]
6. Bargeron, D. and Moscovich, T. 2003. Reflowing digital ink annotations. In Proceedings of CHI '03. pp. 385-393
7. ANSI (1981), *American national standard proof corrections*. 1981: American National Standards Institute.
8. Siger – Simple Gesture Recognizer. http://sourceforge.net/projects/siger/