ProofRite: A Paper-Augmented Word Processor

Kevin Conroy *Dave Levin* *François Guimbretière*

Department of Computer Science
Human-Computer Interaction Lab
University of Maryland,
College Park, MD, 20742
{kmconroy, dml, francois}@cs.umd.edu

ABSTRACT
While proofreading digital documents is a common pattern of use among word processor users [29], at present there are no word processing programs that support this function. This forces users to reenter the corrections into the digital version of a document manually, a time-consuming and error-prone task. To address this problem, we introduce ProofRite, a word processor that supports digital and physical document annotation. When users print a ProofRite document and annotate it with a digital pen, they may merge their changes with the digital source. As they continue the writing process, ProofRite reflows these markings.

ProofRite leverages the features of the first fully implemented PADD infrastructure [11]. This allows ProofRite to serve several common patterns of use: the exchange of paper documents between users from different organizations; the simultaneous annotation of different printed copies; and, the annotation of the same document by different users.

In this paper, we report our experience in designing the first implementation of the PADD infrastructure and how it was used to extend AbiWord into the ProofRite system.

INTRODUCTION
For many users, proofreading a document typically involves printing the document and making annotations, or proofreading marks, with a pen. Although this process provides users with a robust interface for document annotation, it also creates a physical separation between annotations made on paper and the digital source. Consequently, users are forced to type changes and corrections manually, a time-consuming and error-prone process.

This paper introduces ProofRite, a word processor designed to bridge the gap between editing a document on the computer and annotating it on paper. By using a Paper Augmented Digital Document (PADD) infrastructure [11] to record the correspondence between digital and printed versions of a document, ProofRite allows user to merge strokes made on paper using a digital pen with their digital source. Annotations collected by the system are treated as an integral part of the digital document. In particular, strokes reflow with the text with which they have been associated (Figure 1). This flexibility allows users to address annotations and corrections in the order that is most convenient for them without worrying about losing the context of their markings.

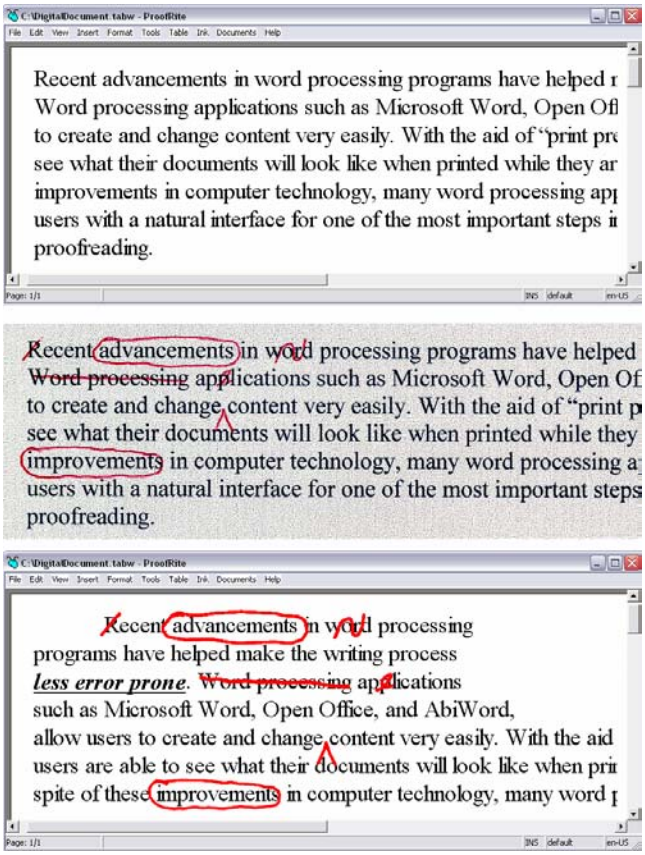

Figure 1: Top: ProofRite document. Middle: Same document printed and annotated on Anoto paper. Bottom: Strokes reflowed in ProofRite.

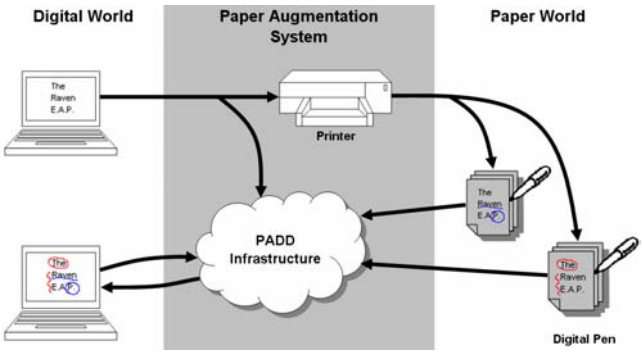

Figure 2: System architecture of ProofRite.

With ProofRite, users are free to edit a document in the format that best fits their needs – on their desktop computer, on paper, or on a Tablet PC – knowing that the information they are gathering will always be readily available within their word processor.

ProofRite is also the first system to leverage the potential of the Paper Augmented Digital Document infrastructure to serve common users needs: such as, the exchange of paper documents between users from different organizations; the simultaneous annotation of different printed copies; and the annotation of the same document by different users. In this paper we discuss how our design of the first fully implemented PADD infrastructure differs from that of [11]. We also propose novel additions to this infrastructure.

MOTIVATION

The gap between the paper and digital worlds has often been a source of frustration for word processor users. Users who prefer to annotate documents on paper are forced to transfer changes to the digital document manually. ProofRite is an extension of AbiWord [1] that addresses this problem by creating a bridge between digital documents and their printouts.

An example of how ProofRite can be used is shown in how Alice and Bob, two knowledge workers from two different universities, use ProofRite to edit a paper on which they are collaborating for an upcoming conference: Once the first draft is ready, Alice and Bob decide to meet. Before the meeting, Alice uses ProofRite to print two copies of the current draft that she is planning to give Bob for review. During the meeting, Alice presents her changes to Bob and uses her digital pen to annotate the documents with notes. Bob also take notes and agrees to review the document in more detail later that day. Alice comes back to her desk and synchronizes her digital pen. Immediately, the document she used in the meeting automatically appears on the screen. Alice may then have the system download all the annotations and, subsequently, see Bob and her own notes overlaid simultaneously on the screen in different colors. The proofreading cycle is completed as annotations reflow, while Alice continues the writing process by making changes to the document's content, layout, or structure.

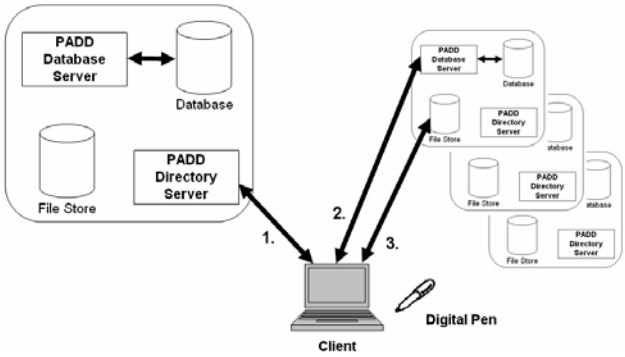

Figure 3: The communication between PADD client and infrastructure. The client (1) contacts the local directory server to determine the appropriate PADD Database, (2) contacts the PADD Database server, and (3) uploads or downloads files from the File Store.

In the remainder of this paper, we describe the main components of the system, which allow users to transfer annotations from paper to digital sources. We explain how we have expanded the PADD infrastructure to support complex use cases, such as the one described above. We, then, present a method to support these annotations digitally by allowing users to reflow markings in the ProofRite word processor. Finally, we evaluate our system and conclude with a discussion of future work.

PROOFRITE

The ProofRite system is divided into two major components. The first component is the Paper Augmented Digital Document infrastructure which provides support for establishing and maintaining the link between digital documents and their printouts. The second part is an extension of AbiWord [1] that interacts with the PADD infrastructure, to fetch the strokes made on paper and then, merge them into the digital documents, as shown in Figure 2. Upon merging, the strokes become part of the document's structure. In particular, they reflow as the document changes. As a convenience, our extension also provides a direct inking interface for TabletPC users. The following section presents each part of our system, in turn.

The PADD infrastructure

Paper-Augmented Digital Documents (PADD) were first introduced in [11] as a means to bridge the gap between the digital and paper worlds. While Guimbretière presented a standalone implementation of the infrastructure as an Acrobat plug-in, ours is the first fully distributed implementation of a PADD system. The architecture of our system is presented Figure 3 and consists of the following key elements:

- A **PADD Service Provider (PSP)**, the administrative entities in the PADD infrastructure. PSPs are responsible for hosting ranges of paper page IDs. In Figure 3, PSPs are shown as round-edged boxes.

- A **directory service** which establishes and maintains the relationship between the paper page ID and the PADD service provider managing them,
- A **PADD Database (PADD Database)**, a database service that stores the information necessary to determine, the strokes files and printing information of a document. Infrastructure clients use this information when merging strokes into digital documents.
- A **file store**, which stores a snapshot of printed document as well as their corresponding strokes files. These snapshots assure the users that, when annotations are incorporated into a document, the strokes line up with the document as intended. Note that in [11], the PADD Database and the file store are treated as a single entity called the PADD Database.
- A **digital pen and paper system**, which can sense and capture strokes made on a printout.
- A **Stroke Collector**, which gathers strokes captured by the digital pen and paper system and stores them at the correct PSP.
- A **Stroke Processor**, which, upon notification by the user (or the PADD service Provider), merges a digital document with the strokes captured on its printout. In [11], the Strokes Collector and the Strokes Processor are considered a single entity called the Stroke Collector.

The following section presents the design rationale for each of these entities.

PADD Service Provider

The PADD infrastructure consists of potentially many administrative entities. We refer to these as PADD Service Providers (PSPs). Each PSP is responsible for providing a directory service server, PADD DB, and file store for a range of page IDs.

Directory Service

The directory service establishes the correspondence between a printout on a given page and the PSP managing that page. This service is similar in nature to other lookup protocols, DNS being the most well known example [23]. Our system, however, is posed with the added difficulty of having to provide a lookup for non-hierarchical page addresses (as opposed to hierarchical domain names in DNS). Thus, our system is more closely analogous to reverse DNS lookups.

Ensuring efficiency of lookups will require proper page address distribution to PSPs, with a focus on address aggregation, as suggested with IPv6 [27]. Given that our installed base is still very small (two sites) our current implementation relies on a simple database that is maintained and distributed manually.

As presented in the Discussion section below, the directory service is the clients' main access point for the PADD infrastructure and is somewhat equivalent to the SMTP server for the email system [16]. Once users know the name of their local PADD directory service, they are ready to use the *entire* system without any additional information. As in the case of SMTP, we anticipate that a canonical server name such as PADD.<domain-name> might be used, making the setup process even simpler.

PADD Database

The PADD Database (DB) is an addition to the architecture proposed by [11]. The PADD Database stores all the necessary associations when merging a document and the strokes files collected on its printouts. This includes, for each document, the document store retaining the digital version of the document and, for each printout, the calibration information for this printout (portrait, landscape, scaling, etc.), and, finally, the list of file stores storing the strokes. In a very natural way, PADD DBs are the gatekeepers for document access. In the Discussion section, we describe how one might leverage this fact to yield more functionality from the PADD infrastructure.

While the original PADD description in [11]] considered the document database as a single entity, we decided to split it into a PADD Database and a file store for, as we did not see the coupling as necessary. We discuss the motivations and implications of this further in the File Store section below.

Another difference between our implementation and that of [11] is that ours provides support for multiple users and multiple annotations to documents. This was not possible in the previous design because the document database resided within the document itself, making concurrent, distributed access impossible. In our implementation, we have developed a client-server architecture in which the PADD Database server acts as a front end to a database that runs independently of any other processes. Thus, we have unified all database updates without requiring documents to grow in size. We decided to use MySQL as the underlying database.

The underlying database itself is only lightly coupled to the PADD infrastructure. This way, other software can be created to browse and alter the data easily. To demonstrate this, we have implemented a Web-based tool that allows users to browse document information online, including the number of times a document has been printed or annotated and which users have annotated the document.

File Store

The main function of the file store is to hold the snapshot of the documents before they are printed and to hold strokes files in a place that remote users can access. As explained in [11], this is necessary to guarantee an accurate merging of the information captured on paper with the corresponding digital instance.

While one might consider storing the documents on one's own computer, this poses the problem of outside access. This is similar to the situation of hosting Web sites; users could run their own HTTP server, but they have greater

difficulty guaranteeing uptime than a more established web service provider would.

As mentioned in the PADD Database section above, we decided to separate the original concept of a document database into a PADD Database and file stores. In our design, file stores are coupled with PADD Databases in terms of policy but decoupled in terms of mechanism. In other words, PADD DBs are aware of file stores and know what each of them stores, but are unaware of the specifics of the protocols. This way, PSPs can choose their own methods of storing files – FTP servers, censorship resistant file stores, etc. – without requiring any infrastructure changes. Allowing for such flexibility is beneficial in multiple ways. For one, deployment is made easier, as PSPs may make use of existing file stores, assuming their clients are aware of the protocol. Secondly, with regards to security, PSPs may offer varying degrees of security (SFTP versus FTP, for example) without any infrastructure changes. Security is discussed further in the Discussion section.

In general, we anticipate that the storage need of the system's users could be different. Some users might not want to store their documents outside their own company; others users might require that their document be encrypted; still others might wish to store their document using a private protocol. By splitting the document database into two entities, we are providing the users with the freedom to address their storage requirement as they see fit.

Digital Pen and Paper System

Like [11], our system uses the Anoto digital paper technology [2]. The Anoto system is based on printing a fine grid of dots on top of normal paper. To the naked eye, this pattern looks like a light grey background, but when observed by a digital pen such as the Logitech io Digital Pen, it provides enough information to capture not only the position of the pen on the page, but also a unique page ID. The pattern space managed by Anoto is very large (more than 2^{48} pages) which makes it easy to (for the foreseeable future) establish a one to one correspondence between a given page in a digital document and a given printed page. It is worth noting that our system does not require the use of the Anoto system, but it is the only currently available system.

Stroke Collector

The stroke collector is the piece of software that detects when a client has synchronized their digital pen and uploads the strokes to the proper PSP. This differs from the stroke collector in [11] which performed the function of the stroke processor, below.

Stroke Processor

The stroke processor is an analogue to the stroke collector in [11]. Given a digital document and a set of strokes files, it is responsible for properly merging them into a single digital document.

We decided to split the original stroke collector into processor and collector because, in multi-user environments, it is often the case that a computer receiving strokes is not the same computer that is incorporating them into a document. In our example, Bob may not have any desire to incorporate his annotations into a document, so his stroke collector need only upload them so that Alice's stroke processor may have access to them.

Client Interfaces API

To simplify the development of new PADD client, we developed a client API, which simplify the integration of the PADD infrastructure. Our API was developed in C++ and has been used not only for ProofRite (see below), but also in developing a new Acrobat plug-in that leverages the benefit of the distributed PADD infrastructure. Although our API is easy to use, it requires that each client follow the following guidelines:

- The editor must obtain document ID's from our infrastructure and store them in the digital document themselves.
- The editor must also be capable of incorporating strokes from a strokes file into the digital document.
- Lastly, the editor must be capable of reporting the addresses of the pages on which it prints. This introduces issues of synchronization among editors. Alternatives to this requirement are discussed in Future Work.

With the exception of the final requirement, the document editor is the clear place to provide the above functionality.

Processing strokes

In this section we discuss how ProofRite interacts with the PADD infrastructure described above to allow users to print, annotate, recover, and reflow strokes as they continue the writing process.

Printing and Calibration

When a user goes from editing a document on the computer to annotating the document on paper, they must first print the document. When this occurs, the system obtains and stores a unique document ID, saves a copy of the document, and reports printer-specific calibration information, such as page size and scaling level back to the PADD infrastructure. This information is later used to recreate and incorporate physical strokes into the digital document.

Recovering the strokes

When Alice wishes to review the comments made by Bob and Carol, she must recover the strokes uploaded to the PADD infrastructure. The process of recovering strokes from paper can be viewed as synchronizing the layer of ink in the digital document with the ink on the paper document, as seen in Figure 5.

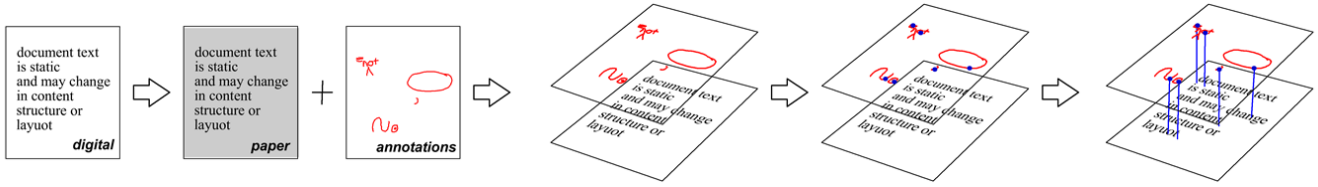

Figure 5: (1) Marks made on paper can be (2) viewed as a separate layer. To synchronize these layers, (3) we identify the semantic insertion point of the mark and (4) insert an anchor at that point, creating a multivalent layer of ink.

To do this, the ProofRite system queries the PADD infrastructure for all strokes made on a particular document. The association between the unique document ID and the unique page address allows the PADD infrastructure to determine which strokes go with which digital documents.

Once the PADD system returns the collection of strokes made on the printed copy, the ProofRite system converts stroke data from the digital pen format to a digital ink format, which is capable of drawing strokes using colored, anti-aliased Bezier curves with pressure information. This allows the ProofRite system to more accurately reproduce the look-and-feel of a real pen stroke on the computer. In our case, we convert strokes from the Logitech io digital stroke format to the Microsoft TabletPC API stroke format. Once converted, ProofRite uses the printing calibration information to compute the affine transform from paper space to document space.

Recovering stroke data from the TabletPC stylus is less complex, as the stroke data is reported in screen coordinates. A simple transformation into the editor-specific coordinate system allows the stroke data to be integrated with the document.

Identifying target text

When an annotation is made on the paper or digital document and later repositioned in the digital copy, the semantic meaning of the mark must be preserved. This is particularly important for proofreading annotations as their meaning is dependent upon the location at which they are displayed.

For this prototype we focus on the ANSI standard proofreading marks [3] which are small and whose semantic meaning typically only corresponds to a single letter or word. However, these marks can be drawn over several characters, words, or lines. For example, consider the upside down "v" insertion mark as seen in Figure 6. This mark typically spans one to two lines of text, but corresponds to only a single point in the document.

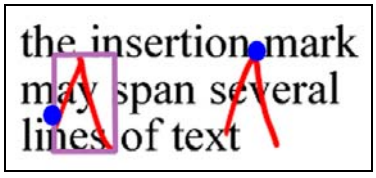

Figure 6: Left: The bounding box method associates the mark with the lines that it overlaps. Right: The change in direction associates the mark at the position of semantic meaning (blue dot).

Existing methods for ink-text association use the bounding box of the stroke. Using such a method for proofreading marks may cause markings which span several lines, particularly the insertion mark, to be associated with text to which the marking has no semantic correspondence.

It is important to note that many proofreading marks are anchored to the text at either end of the marking or at an abrupt change in direction in the mark, as seen in Figure 7. Using this heuristic, our system identifies the point to best insert the stroke. We have found this method of association is sufficient to preserve the semantic meaning of most proofreading marks as the document reflows.

Reflowing annotations

The fundamental challenge of digital reflow is to determine the correct position at which a stroke should be rendered such that the semantic meaning of the mark is preserved.

Reflow algorithms in previous systems have required that the document's content remain static [10]. This constraint simplifies the issues of reflow as the system only needs to locate the new position at which the corresponding text is rendered and apply the appropriate affine transformation to the stroke to move it to this position.

ProofRite is unique from previous systems in that it supports reflow in documents with non-static text. For instance, in ProofRite a user may circle a word, change the

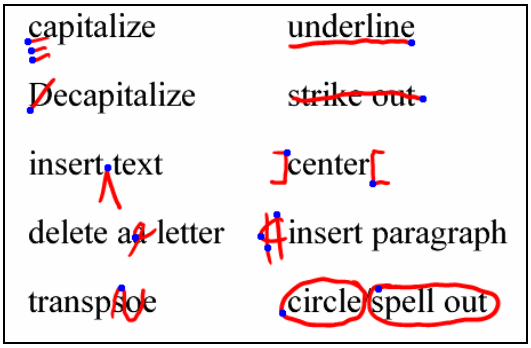

Figure 7: Common proofreading marks and their corresponding insertion point (blue dots).

text on both sides of the mark, change to a different font, and even change the structure of the document by indenting the paragraph without losing the semantic meaning of the mark (Figure 8). This is achieved through the use of intra-document anchors which do not rely on the surrounding text to preserve ink-text association, similar to [26]. This allows the document to be modified in an arbitrarily complex manner.

We leverage the capabilities of the AbiWord rendering engine to compute the position at which an intra-document stroke anchor should be displayed. Rather than drawing the anchor, the rendering algorithm performs an affine transformation which reflows the stroke and then renders it on the screen. Rendering engines in other word processors can be extended similarly.

Additionally, the rendering algorithm supports scrolling and zooming by applying translation and scaling transformations to the ink layer as the user interacts with the document.

An important issue to consider in reflow is ensuring that that word processor is responsive. AbiWord, like many word processors, uses caching and clipping only to re-render portions of the document that have changed significantly. If a paragraph of text is moved down a line but no changes have been made, then the cached version of the paragraph may be translated down rather than re-rendering the underlying document elements. When this occurs, it is necessary to either translate a cached version of the ink or to redraw the corresponding portion of ink. ProofRite takes the latter approach as there is no noticeable degradation of performance.

In this way, the layer of ink that is above the text of the document is synchronized with any changes made to the underlying document, preserving the association between the annotation and the text.

PREVIOUS WORK

Various systems have considered the issues surrounding document annotation. Some systems have focused on recreating the user experience of annotating a paper document on a computer. Other systems have taken the approach of digitally augmenting paper interfaces in an effort to close the gap between paper and computers.

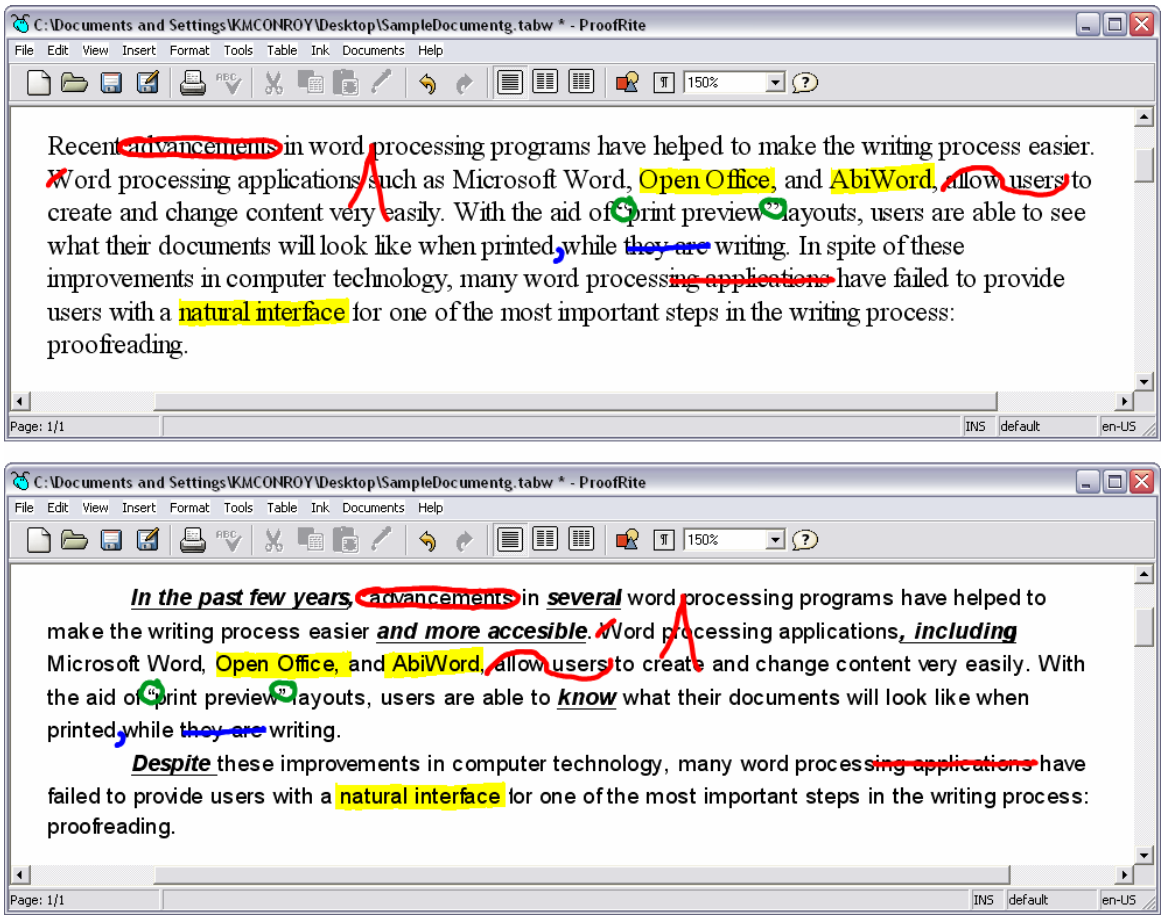

Figure 8: Above: A document with digital annotations. Below: The same document after the user has added text (in underlined-bold-italics for emphasis), deleted some text, created a new paragraph, and changed the font from Times New Roman, size 12 to Arial, size 10.

Annotation and Proofreading

Several systems have explored the issues of digital document annotation. The FreeStyle system [18] and XLibris system [10] introduced an *active reading* system [28] that allows users to digitally annotate a document while reading. Brush et. al. [7] studied user expectations for digital reflow of annotations and, based on these results, Bageron and Moscovich [5] introduced the Callisto framework. Although XLibris and Callisto allow users to annotate documents digitally and reposition these annotations as the layout changes, neither system permits the user to change the document's contents. ProofRite uses a repositioning algorithm that permits the document text to change, allowing users to make arbitrarily dramatic changes to the document's content and structure while preserving the semantic meaning of annotations.

Phelps and Wilensky's [26] notion of multivalent documents provides a suitable metaphor for the relationship between annotations and the document's text: annotations are logically separate from the document text yet annotations have a semantic meaning with the text that must be preserved as the document changes. The notion of a multivalent layer of ink is complicated by the fact it is also necessary to synchronize the layer with annotations made on paper documents. We present a flexible method to synchronize the multivalent layer of ink with the document text using Phelps and Wilensky's [25] intra-document anchors. The PADD infrastructure discussed in this paper introduces methods to synchronize the paper annotations with the digital source.

More recently, Microsoft Word 2003 [22] introduced a limited interface for making digital ink annotations on the Tablet PC. Ink annotations are only repositioned vertically, causing many markings to change or lose meaning as the document's content or layout is repositioned horizontally. Our system attempts to preserve the semantic meaning of annotations within a word processor by repositioning ink both vertically and horizontally.

Bridging the paper-digital gap

Several systems have explored methods to bridge the gap between the computer and the paper world. Some like the XLibris system [10] focus on creating the equivalent on paper in the digital world. Other like DigitalDesk [32], VideoMosaic [19], Ariel [20], PaperLink [4], Inteligent paper [9], the EnhancedDesk [17] and the A-book [21] have studied how to augment the use of paper with the aid of a nearby computer.

Like the PADD system [11], ProofRite focuses on cohabitation, considering both paper and digital instances of the document. In particular users can interact with paper printouts without the help of a nearby computer (with the exception of the digital pen, of course).

Several systems have been proposed to process information captured on paper including XaX [15], the Paper PDA [13], Anoto [2]. All of these systems focus on processing forms and do not offer the flexibility of our system. As explained above, our system uses the Anoto system as a stroke capture mechanism since it is the only one currently commercially available. Non-commercially available systems include DataGlyphs [12] and MEMO-PEN [24]. Our infrastructure differs from Anoto's in that it does not require the registration of forms before processing, and offers a distributed architecture that has been designed to support typical, everyday uses of paper such as proofreading.

DISCUSSION AND FUTURE WORK

Through the development process of ProofRite we have observed several ways to expand the PADD infrastructure to support complex use cases as well as ways to improve user interaction with digital annotations. In this section we discuss these ideas.

Additions to the PADD infrastructure

The design of the PADD infrastructure, as discussed above, is capable of supporting distributed storage and retrieval of documents and annotations. In this section, we propose additions to the PADD infrastructure that extend its functionality and improve its suitability for global deployment. We have not implemented these features, but discuss what would be necessary to do so.

Printer Server

In our current implementation, a user must ensure that the document editor knows the page address of the next page in the printer. Without a means of synchronizing this information between multiple editors or multiple users, the information they report to the infrastructure may be incorrect. We can remove this synchronization problem by centralizing this information at a printer server. The printer can determine which page address it is printing to (this can be done trivially by having the printer print the Anoto pattern immediately before printing the document [11]), and can therefore act as a reliable source of information to the PADD Databases.

Security

With any multi-user environment, security must be considered. When using the PADD infrastructure, users may want to control access to their documents and/or their annotations. There are several possible methods and levels of granularity through which this may occur.

One method would be to create user accounts on the PSP. Users can log into the PSP and can control access to their documents and annotations by informing the PSP which users may access their documents and annotations. This solution allows users to set permissions for groups of annotations and for documents.

Permission granularity is an important part of security considerations. We introduce the notion of annotation-level permissions. Users may wish, for example, that annotations

made with one digital pen have different access levels from those made with another digital pen. In our example, Bob may have a red pen that he only wants other faculty at his company to be able to see, but a blue pen that any user can access.

Note that, in general, security is an issue that pervades the entire infrastructure, as well as external methods such as file encryption. We have designed our version of the PADD infrastructure in a highly modular way so as to be easily extended to incorporate any future security considerations.

Strokes Processor

Once a user has uploaded strokes to the PADD infrastructure, the user typically wishes to integrate these strokes into the digital document. In [11], the software that performs this integration is referred to as the *strokes processor*. In our implementation, the strokes processor resides within a user's document editor. We propose a more general strokes processor that no longer requires explicit user interaction; rather, processors exist as services which can accept requests from the PADD infrastructure itself.

Strokes processors can be written to perform handwriting recognition in batch or to incorporate strokes into documents. This can make the time-consuming process of integrating a large number of strokes into a large document, such as a book, transparent to the user.

In addition to the security concerns discussed above, users may want varying grades of privacy based on the annotations that they have made. A strokes processor on the user's local machine can filter annotations and apply user-defined security policies. For example, a strokes processor can filter annotations uploaded from Bob's digital pen and determine which annotations should be shared with Alice.

Notifications

While deploying the PADD infrastructure, we have noticed that there are several situations when users or software may wish to be notified: (1) Authors often want to know when someone annotates their documents; (2) Users may also wish to know when their co-authors print the latest version of a document; (3) Strokes processors (as discussed above) may want to know when certain documents have been annotated.

In our implementation, the first situation is solved by emailing the document's author. While email is suitable for user interaction, it may not extend to other situations, particularly for stroke processors that operate in batch mode. Rather than implement a different form of notification for each entity, we abstract the idea of notification to that of publish-subscribe systems. Many such systems have been proposed, such as [31], and the design of the PADD infrastructure lends itself well to them.

Incorporating a publish-subscribe system into the PADD infrastructure can be done as follows: The PADD Database can publish events, as it is the central authority for each page within its page range. Other entities can act as subscribers to these events. For example, authors wishing to know when users have annotated their documents would subscribe to these events.

One of the open issues of notification is the concern of event granularity; how detailed can subscription requests be? For instance, strokes processors may only wish to be notified about documents of a certain file type. A more complex situation may be that user Alice wishes to be notified when Bob annotates page 3 of her document. Therefore, the system must permit the use of rules or filters for every subscription, but to what level of detail? Should these filters exist at the publisher, or should Alice's subscriber program filter them itself? We hope to address these questions in our future work.

Version Management

Consider the case where Alice receives Bob's annotations *after* she made structural changes to the document. To properly incorporate Bob's strokes, Alice's strokes processor must merge with the version of the document that was stored when it was printed. This appears similar to the original problem of the paper-digital gap we have presented, but the important difference is that both versions of the document exist in the digital world. Thus, potential solutions to such a versioning problem exist. For instance, versioning software such as CVS can aid in the management of these versions. Incorporating these seamlessly into the proofreading cycle is a large area of potential future work.

Paper as an indexing mechanism

It has been shown that users can glean an efficient semantic structure from the physical distribution of their documents, be it from a file cabinet or an untidy workspace. With the PADD infrastructure, users need only a single page of a physical document to have access to the entire digital document. Therefore, users can benefit from the semantic structure of their physical data in the digital world [14]. In essence, a user's file cabinet can become their file system.

Duplicate page addresses

The Anoto paper system assigns a unique page address to each physical sheet. There are four fields: bookcase, shelf, book, and page – yielding more than 2^{48} sheets of uniquely addressable paper, a number that, although large, is reachable in a matter of years if globally deployed ([1], [24]). Anoto sells licenses to each sheet of paper in page ranges. The number of unique IDs and the overhead of cost limit the amount of pages an organization, such as a university, can purchase. Therefore, there is a high probability that page address collide.

The PADD infrastructure was built with the initial assumption that all page addresses were unique. Otherwise, when uploading strokes, there is currently no way of determining the corresponding document without requiring

work from the user, such as choosing from a list of documents that have been printed to that same page address.

Unless the paper technology is extended to handle more page addresses, there are two ways we envision of solving the problem: requiring input from the user and/or providing a set of private page addresses, similar to NAT (Network Address Translation) [30]. In order to make the transition between the digital and physical world as seamless as is possible, solutions must minimize the amount of necessary user input.

One method is to include a 'key' portion in a piece of paper, which consists of several paper patterns. When a user is done annotating a piece of paper, they can make a line across this portion of the paper. If the key contains k paper patterns, then the page namespace can be effectively extended to $(2^{48})^k$ unique IDs.

The other type of solution is to provide a set of private page addresses, similar to private IP addresses. In the case of IP, private addressing is made possible by NAT, which provides the translation between private and public. There are two main differences between the assumptions made by NAT and the assumptions a paper-based solution could make:

- NAT makes use of port numbers as a multiplexing key; there is currently no similar entity in paper. One could make the claim that a user's pen could be such a key, but this precludes users' ability to share pens.
- Efforts are made in NAT to keep private addresses from being exposed to hosts outside of the corresponding private realm [6]. A method of providing this occlusion of information is not clear, as users will be printing to and distributing pages with private addresses.

There is currently no clear solution to the problem of duplicate page number. Since the most probable solution will come from increasing the amount of user input, user studies are required to determine the precise extent to which users are willing to do this.

Digital annotations and reflow

The ProofRite word processor allows users to reflow annotations as they continue the writing process. The current implementation can be expanded in several ways to support more complex annotations and user interactions.

Grouping margins notes and drawings

Our method of text-annotation association is sufficiently robust for proofreading markup, but is limited in its ability to address freeform writing and annotations. Tests with our system show that our association algorithm is able to preserve freeform markings and marginal notes vertically but is unable to preserve meaning horizontally. We are currently working on a stroke grouping algorithm similar to the one described in [10] which we believe will solve this problem.

Splitting large annotations

As a document's content or layout changes, text that appears on a single line may be repositioned to span multiple lines. This issue is of particular importance in a document with dynamic content. XLibris [10, 28] addressed this problem by splitting annotations into smaller segments that can be positioned over several lines. We implemented a similar algorithm in an earlier version of ProofRite by splitting large annotations into smaller single-word annotations. This allows the user to change the document's content, structure, or layout in any arbitrary manner while still ensuring that the stroke will retain its semantic meaning. As the ProofRite system has matured, our initial algorithm has proved to be insufficient to preserve semantic meaning. We are currently exploring new splitting algorithms to address this problem.

Automatic stroke interpretations

Although we have not conducted any formal user studies of our system, several early test users have expressed interest in the ability to apply annotated corrections automatically. An "auto-correct" or "apply annotations" feature is limited by the accuracy of handwriting and annotation recognition. In order to implement a robust recognition engine, it is necessary not only to examine the annotation, but also the surrounding text and annotations. It may be possible to provide better annotation recognition by examining the position of the stroke relative to the text (i.e. the cross-out mark goes through text while the underline mark goes underneath text).

Digital Annotation Interfaces

Although reflow is an important aspect of digital annotation, the ability to preserve annotations made on paper in the digital source presents another layer of data with which the user can interact during the writing process. This presents several questions regarding the user experience of interacting with digital annotations. When should markings be applied to the text? In what ways do users wish to interact with their markings (i.e. move, erase, change color, etc)? What is the best way to present annotations from several different sources? Does the presence of annotations from the last proofreading affect the user experience of word processing? Until now, annotations made on paper have always been fixed and static, so users have no experience or prior expectations of how to interact with annotations that reflow. The question of how users wish to interact with digital annotations is fundamental to the writing process. Further studies must be conducted to determine how word processors that support digital annotation and reflow can best serve the user.

CONCLUSION

In this paper, we have addressed the gap that exists between the usage of paper and computers with respect to a common paper-computer activity: proofreading. We

introduced the ProofRite word processor. ProofRite allows users to annotate a document with a Tablet PC stylus and, through the first fully-implemented PADD infrastructure, to incorporate paper-based annotations into their digital documents. Both types of annotation are given the benefits of digital reflow, allowing users to continue the writing process without having to lose or manually place annotations between revisions. Furthermore, we described how annotations made by multiple users to multiple copies of a PADD-compliant document can be integrated into a single digital copy. We also proposed several novel additions to the PADD infrastructure that increase its functionality and further make the paper-digital gap transparent to users.

ACKNOWLEDGEMENTS

The authors wish to thank Dom Lachowicz, Marc Maurer, and the rest of the AbiWord development team for technical assistance; Jim Hollan and Ron Stanonik for their assistance in deploying PADD at UCSD; Samantha Ahalt for comments on drafts of this paper.

REFERENCES

1. AbiWord, http://www.abiword.com. 2004.
2. Anoto, Development Guide for Service Enabled by Anoto Functionality. 2002, Anoto.
3. ANSI*, American national standard proof corrections*. 1981: American National Standards Institute.
4. Arai, T., D. Aust, and S.E. Hudson. PaperLink: a technique for hyperlinking from real paper to electronic content. *Proceedings of CHI'97*, pp. 327 - 334.
5. Bargeron, D. and T. Moscovich. Reflowing digital ink annotations. *Proceedings of CHI'03*, pp. 385 - 393.
6. Bellovin, S. A Technique for Counting NATted Hosts. *Proceedings of Second Internet Measurement Workshop*, pp. 267 - 272.
7. Brush, A.J.B., D. Bargeron, A. Gupta, and J.J. Cadiz. Robust annotation positioning in digital documents. *Proceedings of CHI*, pp. 285 - 292.
8. Chang, B.-W., J.D. Mackinlay, P.T. Zellweger, and T. Igarashi. A negotiation architecture for fluid documents. *Proceedings of UIST'98*, pp. 123 - 132.
9. Dymet, M. and M. Copperman. Intelligent Paper. *Proceedings of EP'98*, pp. 392 - 406.
10. Golovchinsky, G. and L. Denoue. Moving markup: repositioning freeform annotations. *Proceedings of UIST'02*, pp. 21 - 30.
11. Guimbretiere, F. Paper Augmented Digital Documents. *Proceedings of UIST'03*, pp. 51 - 60.
12. Hecht, D.L. Embedded Data Glyph Technology for Hardcopy Digital Documents. *Proceedings of SPIE Color Hard Copy and Graphic Arts III*, pp. 341 - 352.
13. Heiner, J.M., S.E. Hudson, and K. Tanaka. Linking and messaging from real paper in the Paper PDA. *Proceedings of UIST'99*, pp. 179 - 186.
14. Ishii, H. and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proceedings of CHI'97*, pp. 234-241.
15. Johnson, W., H. Jellinek, J. Leigh Klotz, R. Rao, and S.K. Card. Bridging the paper and electronic worlds: the paper user interface. *Proceedings of CHI'93*, pp. 507 - 512.
16. Klensin, J., Simple Mail Transfer Protoco, RFC 2821, April 2001.
17. Koike, H., Y. Sato, Y. Kobayashi, H. Tobita, and M. Kobayashi. Interactive textbook and interactive Venn diagram: natural and intuitive interfaces on augmented desk system. *Proceedings of CHI'00*, pp. 121 - 128.
18. Levine, S.R. and S.F. Ehrlich, The Freestyle System: A Design Perspective, in *Human-Machine Interactive Systems*, A. Klinger, Editor. 1991. p. 3-21.
19. Mackay, W. and D. Pagani. Video mosaic: laying out time in a physical space. *Proceedings of MM'94*, pp. 165 - 172.
20. Mackay, W.E., D.S. Pagani, L. Faber, B. Inwood, P. Launiainen, L. Brenta, and V. Pouzol. Ariel: augmenting paper engineering drawings. *Proceedings of CHI'95*, pp. 421 - 422.
21. Mackay, W.E., G. Pothier, C. Letondal, K. Bøegh, and H.E. Sørensen. The missing link: augmenting biology laboratory notebooks. *Proceedings of UIST'02*, pp. 41 - 50.
22. Microsoft, http://www.microsoft.com/word. 2003.
23. Mockapetris, P., Domain Names - Concepts and Facilities, RFC 1034, ISI, November 1987.
24. Nabeshima, S., S. Yamamoto, K. Agusa, and T. Taguchi. MEMO-PEN: a new input device. *Proceedings of CHI'95*, pp. 256 - 257.
25. Phelps, T. and R. Wilensky, Robust intra-document locations. Computer Networks, 2000. **33**(1-6): p. 105 - 118.
26. Phelps, T.A. and R. Wilensky. Multivalent Annotations. *Proceedings of ECDL'97*, pp. 287 - 303.
27. Rekhter, Y. and T. Li, An Architecture for IPv6 Unicast Address Allocation, RFC 1887, Cisco Systems, December 1995.
28. Schilit, B.N., G. Golovchinsky, and M.N. Price. Beyond paper: supporting active reading with free form digital ink annotations. *Proceedings of CHI'98*, pp. 249 - 256.
29. Sellen, A.J. and R.H.R. Harper*, The Myth of the Paperless Office*. 1st ed. 2001: MIT press.
30. Srisuresh, P. and K. Egevang, Traditional IP network address translator (traditional NAT), RFC 3022, Internet Engineering Task Force, January 2001.
31. Stoica, I., D. Adkins, S. Zhaung, S. Shenker, and S. Surana. Internet indirection infrastructure. *Proceedings of SIGCOMM'02*, pp. 73-86.
32. Wellner, P., Interacting with paper on the DigitalDesk. Communications of the ACM, 1993. 36(7): p. 87 - 96.