Freeform Digital Ink Annotations in Electronic Documents: A Systematic Mapping Study

Craig J. Sutherland[a,*], Andrew Luxton-Reilly[a], Beryl Plimmer[a],

[a] *Department of Computer Science, University of Auckland, Private Bag 92019, Auckland 1142,New Zealand*

Abstract

A variety of different approaches have been used to add digital ink annotations to text-based documents. While the majority of research in this field has focused on annotation support for static documents, a small number of studies have investigated support for documents in which the underlying content is changed. Although the approaches used to annotate static documents have been relatively successful, the annotation of dynamic text documents poses significant challenges which remain largely unsolved. However, it is difficult to clearly identify the successful techniques and the remaining challenges since there has not yet been a comprehensive review of digital ink annotation research. This paper reports the results of a systematic mapping study of existing work, and presents a taxonomy categorizing digital ink annotation research.

Keywords: Freeform Ink Annotation, Dynamic Digital Documents

1. Introduction

Handwritten annotations are an easy and effective way to actively engage with a document [2, 53] that is shown to improve comprehension and retention [7, 73]. Early research to support annotation of digital documents focused on implementing text-based annotations in which annotations were added using a mouse and keyboard [53]. Modern pen and touch input devices

*Corresponding author

Email addresses: cj.sutherland@auckland.ac.nz (Craig J. Sutherland), andrew@cs.auckland.ac.nz (Andrew Luxton-Reilly), beryl@cs.auckland.ac.nz (Beryl Plimmer)

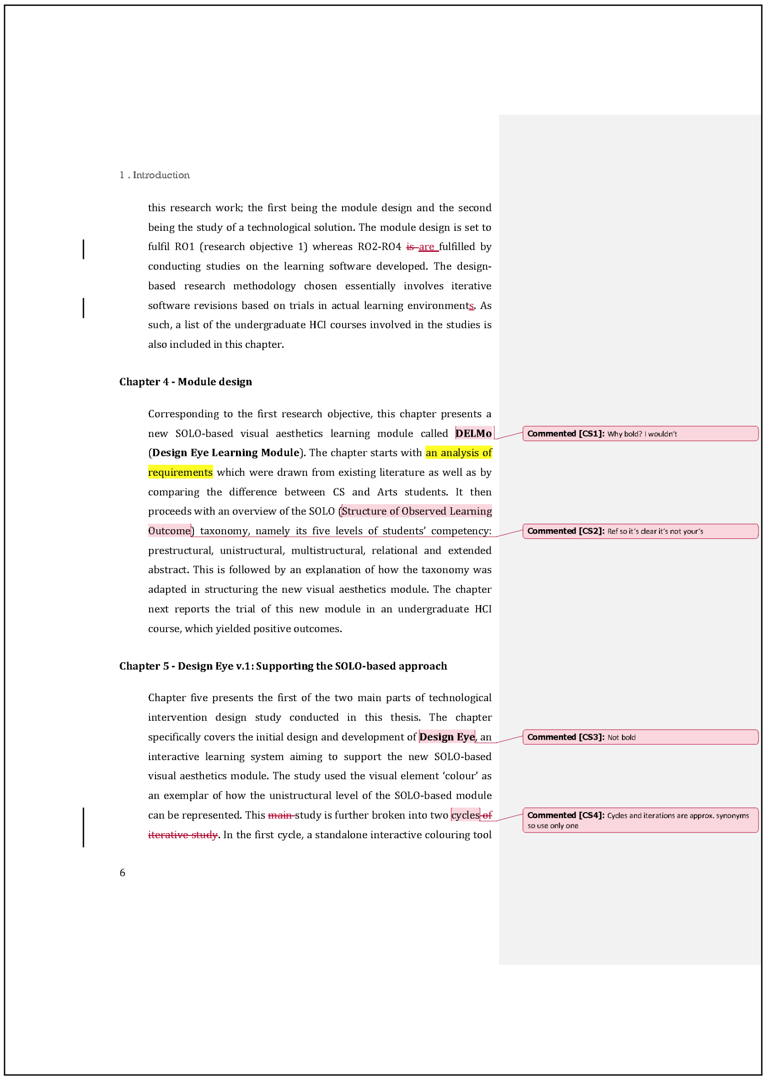
allow for freeform digital ink annotations similar to pen on paper [52]. There are numerous approaches that support digital ink annotations on static documents but annotating dynamic documents poses significant technical challenges that remain unsolved. Yet a core advantage of most digital document formats is their inherent support for change. It is difficult to clearly identify the successful techniques for dynamic annotation support and distinguish them from the remaining challenges since there has not yet been a comprehensive review of digital ink annotation research. In this literature review we develop and apply a taxonomy to categorize current research, and report on the solutions and remaining challenges.

The impetus for ink annotation support in digital tools is that freeform annotations offer two benefits over text annotations. First, annotating with a pen is less cognitively demanding than with a mouse and keyboard [53]. Second, ink annotations stand out from the underlying text and are easier to find [53].

Adding annotations using a keyboard and mouse requires a higher mental workload than using a pen. The user has to switch from thinking about what they are reading to how they are going to annotate [53]. This mental switch is increased by software implementations that require the reader to annotate in a different location from where they are reading. As a consequence people annotate less on a computer screen than on paper [43]. Also the increased mental workload of text annotation reduces how much the reader can comprehend and learn [34].

As well as being more cognitively demanding, it is harder to find text annotations than freeform annotations [53]. One important role of annotations is to act as signposts to important information [23]. Consider Figures 1 and 2 , the ink annotations stand out from the page making it easy to see them when scanning through a document. In contrast, even with colour coding, the text annotations blend with the text forcing the reader to spend more time looking for them and less time on comprehension.

While the benefits of freeform ink annotations are widely recognised [34, 52, 67] there are a number of technical challenges involved in adding ink annotations to digital text documents. Computers can process the data in text documents due to the structure of the documents. A document consist of a string of characters which is grouped into words, sentences, paragraphs and so on. In contrast freeform annotations lack intrinsic structure. While it is possible to treat annotations as images on a document this limits functionality available. Instead, an application needs some way to unlock the


1 . Introduction

this research work; the first being the module design and the second being the study of a technological solution. The module design is set to fulfil RO1 (research objective 1) whereas RO2-RO4 ~~is~~ are fulfilled by conducting studies on the learning software developed. The design-based research methodology chosen essentially involves iterative software revisions based on trials in actual learning environments. As such, a list of the undergraduate HCI courses involved in the studies is also included in this chapter.

Chapter 4 - Module design

Corresponding to the first research objective, this chapter presents a new SOLO-based visual aesthetics learning module called **DELMo** (**Design Eye Learning Module**). The chapter starts with an analysis of requirements which were drawn from existing literature as well as by comparing the difference between CS and Arts students. It then proceeds with an overview of the SOLO (Structure of Observed Learning Outcome) taxonomy, namely its five levels of students' competency: prestructural, unistructural, multistructural, relational and extended abstract. This is followed by an explanation of how the taxonomy was adapted in structuring the new visual aesthetics module. The chapter next reports the trial of this new module in an undergraduate HCI course, which yielded positive outcomes.

Chapter 5 - Design Eye v.1: Supporting the SOLO-based approach

Chapter five presents the first of the two main parts of technological intervention design study conducted in this thesis. The chapter specifically covers the initial design and development of **Design Eye**, an interactive learning system aiming to support the new SOLO-based visual aesthetics module. The study used the visual element 'colour' as an exemplar of how the unistructural level of the SOLO-based module can be represented. This ~~main~~ study is further broken into two cycles ~~of iterative study~~. In the first cycle, a standalone interactive colouring tool

6

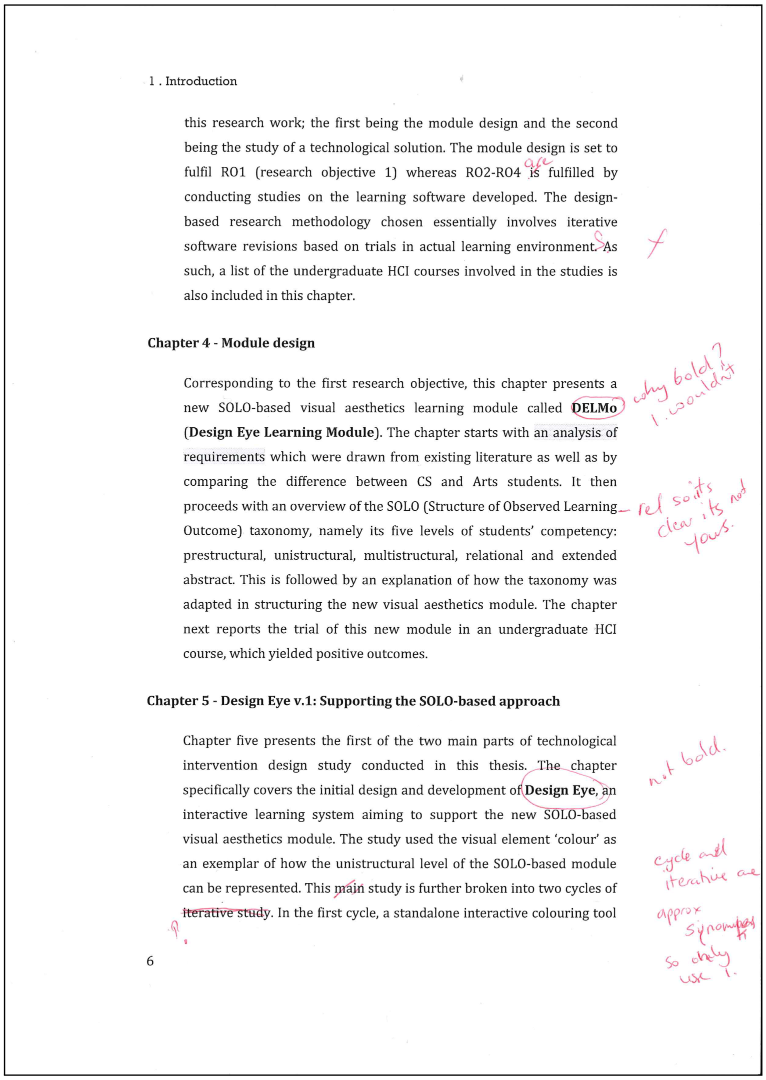
Commented [CS1]: Why bold? I wouldn't

Commented [CS2]: Ref so it's clear it's not your's

Commented [CS3]: Not bold

Commented [CS4]: Cycles and iterations are approx. synonyms so use only one

Figure 1: Examples of text annotations: text-based annotations in Microsoft Word.



1 . Introduction

this research work; the first being the module design and the second being the study of a technological solution. The module design is set to fulfil RO1 (research objective 1) whereas RO2-RO4 is fulfilled by conducting studies on the learning software developed. The design-based research methodology chosen essentially involves iterative software revisions based on trials in actual learning environment. As such, a list of the undergraduate HCI courses involved in the studies is also included in this chapter.

Chapter 4 - Module design

Corresponding to the first research objective, this chapter presents a new SOLO-based visual aesthetics learning module called **DELMo** (**Design Eye Learning Module**). The chapter starts with an analysis of requirements which were drawn from existing literature as well as by comparing the difference between CS and Arts students. It then proceeds with an overview of the SOLO (Structure of Observed Learning Outcome) taxonomy, namely its five levels of students' competency: prestructural, unistructural, multistructural, relational and extended abstract. This is followed by an explanation of how the taxonomy was adapted in structuring the new visual aesthetics module. The chapter next reports the trial of this new module in an undergraduate HCI course, which yielded positive outcomes.

Chapter 5 - Design Eye v.1: Supporting the SOLO-based approach

Chapter five presents the first of the two main parts of technological intervention design study conducted in this thesis. The chapter specifically covers the initial design and development of **Design Eye**, an interactive learning system aiming to support the new SOLO-based visual aesthetics module. The study used the visual element 'colour' as an exemplar of how the unistructural level of the SOLO-based module can be represented. This main study is further broken into two cycles of iterative study. In the first cycle, a standalone interactive colouring tool

6

Figure 2: Examples of text annotations (cont.): digital ink annotations.

implicit data within an annotation.

Converting a document into a static image, and positioning the annotation using standard Cartesian coordinates [55, 67] simplifies the process of adding annotations. However it prevents the document from being edited, eliminating one of the major advantages of digital documents over paper.

Substantive documents (e.g. academic papers, legal contracts) often go through multiple drafts. While reviewing a draft, annotations are used to add questions and suggestions, correct errors, and so forth [1, 69]. Then the draft is updated resulting in changes to the content and structure. The challenge is to adapt the ink annotations as the underlying document changes.

While several projects have explored ink annotation support for dynamic text documents there is not yet a complete solution. The challenges for supporting dynamic ink annotations fall into three broad categories: solving the technical challenges; understanding how people interact with their annotations; and exploring how computers can extend what is possible on paper. In this paper we are interested in how the technical challenges have been approached.

There has not been a comprehensive review of digital ink annotation research. Therefore it is difficult to see what techniques are successful and what challenges are unsolved. We have performed a systematic mapping study and reviewed the published research. There is always a need to draw a boundary in a literature review. We have decided to focus on freeform digital ink annotations in text-based documents. This means there are some publications just outside the boundary line. We have included some examples of these in §2.4 to guide readers to other surrounding fields.

We propose a taxonomy of annotation research characterised by: the types of annotations supported; the operations used to add freeform ink annotations; the operations to automatically adapt freeform annotations on dynamic documents; and the human perception of digital ink annotations. One of the challenges of this review is the plethora of terms inconsistently applied. We refer the reader to the set of terms we propose in §4.1 as these are used through-out the review.

2. Research Method

Systematic literature reviews originated in the field of medical studies. They are used to perform a systematic, comprehensive and reproducible analysis of the research about a given topic. They have been applied in other fields such as software engineering, social sciences, chemistry and education [35]. A systematic mapping study is a variation that provides a wider overview of a research area. They are useful for identifying what has and has not been researched [35, 54]. The attributes of this protocol make it ideally suited to our purposes of documenting and synthesising research on digital ink annotation.

In the systematic mapping study reported in this paper, we use the protocol described by Okoli and Schabram [54], which involves the following steps:

1. Specify the research questions;
2. Protocol development;
3. Review protocol;
4. Study search;
5. Primary studies selection;
6. Data extraction;
7. Data synthesis;

As this is a systematic mapping study, rather than a systematic literature review, we have not included any analysis of quality [36].

Each step is fully documented to ensure that the study can be reproduced.

2.1. Research Questions (Step 1)

The research questions for this study represent the two main challenges in the field. These challenges, identified in earlier research, are adding digital ink to a document and supporting dynamic digital documents [8, 24]. This mapping study is framed by the following research questions:

- RQ1. What methods are described in existing literature for adding freeform annotations to digital documents?
  - RQ1.1 How is digital ink collected?
  - RQ2.2 What is the process for associating an annotation to a location in the document?
  - RQ3.3 What types of annotations are recognised?
- RQ2. What support for automatic adaptations to annotations have been explored to handle changes in the underlying document?
  - RQ2.1 What operations are required to automatic adapt annotations?
  - RQ2.2 How can annotations be automatically adapted?
  - RQ2.3 How have user expectations on automatic adaptations of annotations been studied?

2.2. Protocol Development and Review (Step 2 and 3)

We analysed five publications used in a previous study [77] to identify potential key terms. These terms were used to define the primary search string and possible alternatives.

A data extraction form was also developed. This form listed the data items to obtain from each publication. These items were chosen based on the research questions. Some of these items are selections with the initial values based on values from the initial five publications.

During the development of the protocol the data extraction form was trialled in a small group. The trial evaluated the definitions of each item to ensure consistency. One of the publications from the previous study was used [24]. Each member extracted all the data items. Based on the feedback the data extraction form was modified to clarify the definitions of each item. The final form is shown in Table 1.

2.3. Search Strategy (Step 4)

The search string, Annotation AND "Digital Ink", was trialled on the following six databases:

1. ACM Digital Library;
2. IEEE Xplore;
3. SpringerLink;
4. Scopus;
5. Inspec;
6. ProQuest.

The search string found all five of the initial publications. Some alternate search strings were also tried but these either did not add any additional results or were too broad.

After the search string was finalised the search was run on all six databases on a single day (24^{th} December, 2013). The databased were searched in the order listed above. Where possible a full text search was used, otherwise the fullest search options were used.

During the search the results were extracted into a table. The information recorded in the table for each publication included:

(i) Year of publication;
(ii) Venue;
(iii) Authors;

(iv) Title;

(v) Source.

Duplicate results from multiple databases were added to the table. When there were duplicate results within a database (e.g. a conference proceeding and journal article for the same study) the most recent one was used.

After the initial set of publications were selected (Step 5) we returned to this step and performed a forwards and backwards search using the references and citations of each selected publication. After this search we then re-applied step 5 to the new results. We only performed one iteration of forward and backwards searching.

2.4. Selection Criteria (Step 5)

We selected the final publications using multiple phases. In the first phase we checked the source details of each publication. Non-peer reviewed publications (e.g. magazine articles) and non-English publications were excluded. We also identified duplicates based on the authors, date and title and excluded all duplicates but the most recent.

In the second phase we excluded publications whose topic was out of scope. We are specifically interested in the annotation of text-based documents. Therefore publications examining annotation of video and audio files were excluded as they are not text-based (e.g. [12]). Whiteboard applications were excluded for the same reason (e.g. [10]). Drawing and sketching applications were excluded as they start with a blank canvas rather than a pre-existing document (e.g. [87]). Finally implementations with text-only annotations were excluded as we are specifically interested in freeform digital ink (e.g. [88]). The exclusion criteria were applied using the publication's title and abstract. If there was any doubt about whether a publication should be excluded it was left in. When a publication was excluded the reason for exclusion was noted in the table.

For the next phase the full-text of each remaining publication was retrieved. If a publication could not be retrieved in this phase it was excluded. An example of a publication that could not be retrieved is one that is marked as a citation in the search engine without a link or DOI to retrieve it. With these publications we made all possible attempts to retrieve them including using mutliple libraries and other search engines.

For the final phase, the abstract and conclusion of each publication was checked to ensure it met the following inclusion criteria:

Table 1: Data extraction form

	Name	Type
General Data	Year of Publication	Numeric
	Authors	Free text
	Title	Free text
	Publication venue (e.g. conference or journal name)	Free text
	Abstract	Free text
	Name of implementation	Free text
System Data	Overview	Free text
	Input mechanism	Selection
	Application Domain	Free text
	Document type	Selection
	Overview of adding annotations	Free text
	Annotation types recognized	Selection
	Overview of adapting annotations	Free text
	Change type supported	Selection
	Usability study results	Free text
	Additional functionality provided	Free text

(i) The publication must include an implementation of a system;
(ii) The implementation must allow users to add annotations;
(iii) The implementation must use digital ink;
(iv) The annotations must be for a document (e.g. not a blank notebook).

If there was insufficient detail in the abstract and conclusion to determine whether to include the publication, the rest of the publication was scanned. Any publication that did not meet the inclusion criteria was excluded.

2.5. Data Extraction (Step 6)

The final list of publications was analysed using the data extraction form (see Table 1) to collect the details on each publication. This form has two main sections: general data and system data. The items in the general data section are based on the systematic mapping protocol [54]. The items in the system data section, described below, are based on the research questions.

2.5.1. Overview

This is a summary of the publication, it includes what the implementation was attempting to do, what was actually achieved and what was involved.

2.5.2. Input mechanism

Input mechanism is how the ink was physically collected. This started off as two options: tablet with stylus and Anoto pen on paper. As we found additional input mechanisms the list was expanded.

2.5.3. Application domain

Application domain is the target domain of the application. This is a free text field to allow for any options. During the data synthesis this list was consolidated (see §2.6).

2.5.4. Document type

The *document type* describes the format of the text-based document. Based on the initial protocol development the starting values for this were 'text only', 'Word' and 'PDF'. As additional formats were found they were added to the list of values.

2.5.5. Overview of adding annotations

Adding annotations is the process of collecting digital ink strokes and associating them with the document. This overview lists the reported details on how an implementation handled adding annotations.

2.5.6. Annotation types recognised

An *annotation type* is a class of annotations that shares similar properties. For example underlines are lines drawn underneath text. Marshall [46] proposed a number of annotation types and her types were used as the initial values. If a new type was found during the extraction phase it was added to the list of types. During the data synthesis this list was consolidated (see §2.6).

2.5.7. Overview of adapting annotations

Adapting annotations is the process of automatically modifying an annotation in response to a change in the document. This overview lists the reported details on any adaptations performed by the implementation.

2.5.8. Change type supported

Change type supported defines how the underlying document could change. This selection is based on the spectrum proposed by Golovchinsky and Denoue [24]: none; layout-only; and layout-and-content.

None means the implementation does not handle any changes to the document. This assumes that the document remains static through-out the lifetime of the annotations. The PDF format is an example of a format that does not allow changes[1]. *Layout-only* means the rendering of the document can change but not the content. Examples of layout changes include changing the font size, page margin, zoom factor, etc. The ePub format is a format that allows for layout changes. *Layout-and-content* means that both the layout and the content of the document can change. For example text can be added, modified or deleted, images and other objects can be inserted or removed. Plain text is an example format for this change type. We were unable to determine what types of changes are supported in some implementations. These implementations were recorded as unknown.

2.5.9. Usability study results

These are the details of any human studies reported in the publication.

2.5.10. Additional functionality provided

This lists any additional functionality provided in the implementation. This functionality extends what is available using just pen and paper. For example, XLibris explored how annotations could be used to search search queries [27].

2.6. Data Synthesis (Step 7)

After data extraction the publications were grouped by implementation. Details were merged together when there were multiple publications about a single implementation. For the *annotation types recognised* field we included the values from all publications. For the *input mechanism* and *change type supported* fields we used the values from the most recent publication. For the remaining fields we compared the publications. If there were details mentioned in one publication but not another they were combined. If there

[1]There are now tools that allow changing a PDF document but the original intention of the specification is for read-only documents.

were conflicting details the details from the most recent publication were used.

The list of *annotation types recognised* was consolidated. Some publications used different terms for the same type of annotation (e.g. circling, enclosure and box are all synonymous). Each term was checked to see if they referred to the same *annotation type*. If so they were combined together and the most common term used.

Next, the *addition* and *adaptation process* overviews were analysed. From each overview all the operations that were mentioned were listed. For example, for addition a publication might mention "combining strokes" and "linking to underlying context". This produced one list of operations for adding an annotation to the document and a second list of operations for adapting annotations.

The lists were then reviewed to find operations that were similar. For example "combining strokes" and "grouping strokes" both refer to the same operation. Grouping similar operations together produced one set of operations for adding annotations and another for adapting them. To check the completeness of each set all the implementations were reviewed to determine which operations where implemented.

During this process we discovered the mode of anchoring an annotation to the underlying document is important in adding annotations. So the data synthesis step was repeated to capture the different anchor modes.

Finally, the results from *input mechanisms*, *application domains* and *additional functionality* were consolidated. Each item was summarised in a few words that described the main feature (e.g. "Presenting and annotating slides" was summarised as "Lecture Presentation"). These summaries were then grouped together where possible. If the summaries referred to the same feature then they were combined (e.g. "Writing documents" and "Editing documents" were combined). Finally the summaries were grouped into relevant hierarchies for application domains and additional functionality.

3. Results

3.1. Search Results

A total of 801 publications were found during the initial search phase. The selection process described in step 5 of the methodology was then applied (see §2.4). Out of the 801 publications 48 met the inclusion criteria for the study. Using these publications we performed the backwards and forwards search and found an additional 578 publications. Again, we checked these publications against the exclusion and inclusion criteria and identified a further 13 publications to include.

A total of 61 publications were included in the mapping study. These publications describe 42 different implementations. Most implementations are described by a single publication. Six implementations have two publications(CodeAnnotator, CoScribe, OneNote, Papiercraft, PenMarked and United slates), three implementations have three publications (Classroom Presenter, RCA and WriteOn) and XLibris has ten publications.

Table 2: Publications that were included in the mapping study.

Year	Authors	Implementation	Input Mechanism	Change Allowed	Application Domain
1991	Levine and Ehrlich	FreeStyle	Digitizer	None	Collaboration
1993	Hardock, Kurtenbach, and Buxton	MATE	Digitizer	Content	Editing documents
1998	Price, Golovchinsky, and Schilit	Xlibris	Tablet PC	None	Active reading
1998	Schilit, Golovchinsky, and Price	Xlibris	Tablet PC	None	Active reading
1998	Schilit, Price, and Golovchinsky	Xlibris	Tablet PC	None	Active reading
1999	Golovchinsky, Price, and Schilit	Xlibris	Tablet PC	None	Active reading
1999	Marshall, Price, Golovchinsky, and Schilit	Xlibris	Tablet PC	None	Active reading
1999	Truong, Abowd, and Brotherton	Classroom 2000	Tablet PC	None	Lecture presentation
2000	Golovchinsky and Marshall	Xlibris	Tablet PC	None	Active reading
2000	Golovchinsky and Marshall	Xlibris	Tablet PC	None	Active reading
2001	Marshall, Price, Golovchinsky, and Schilit	Xlibris	Tablet PC	None	Active reading
2002	Golovchinsky and Denoue	Xlibris	Tablet PC	Layout	Active reading
2002	Götze, Schlechtweg, and Strothotte	Intelligent pen	Unknown	None	Active reading
2002	Mackay, Pothier, Letondal, Bøegh, and Sørensen	A-book	Multiple	None	Biology lab
2003	Bargeron and Moscovich	Callisto	Tablet PC	Content	Not specified
2003	Guimbretière	PADD	Anoto	None	Not specified
2003	Ramachandran and Kashi	Ramachandran & Kashi (2003)	Unknown	Content	Web browsing
2003	Shipman, Price, Marshall, and Golovchinsky	Xlibris	Tablet PC	None	Active reading

Table 2: (continued).

Year	Authors	Implementation	Input Mechanism	Change Allowed	Application Domain
2004	Anderson, Anderson, Simon, Wolfman, VanDeGrift, and Yasuhara	Classroom Presenter	Tablet PC	None	Lecture presentation
2004	Anderson, Hoyer, Prince, Su, Videon, and Wolfman	Classroom Presenter	Tablet PC	None	Lecture presentation
2004	Conroy, Levin, and Guimbretière	ProofRite	Multiple	Content	Editing documents
2004	Olsen, Taufer, and Fails	ScreenCrayons	Tablet PC	None	Not specified
2004	Shilman and Wei	Shilman & Wei (2004)	Tablet PC	Unknown	Not specified
2005	Agrawala and Shilman	DIZI	Tablet PC	None	Not specified
2005	Anderson, McDowell, and Simon	Classroom Presenter	Tablet PC	None	Lecture presentation
2005	Dontcheva, Drucker, and Cohen	v4v	Tablet PC	None	Lecture presentation
2005	Kam, Wang, Iles, Tse, Chiu, Glaser, Tarshish, and Canny	LiveNotes	Tablet PC	None	Lecture presentation
2005	Liao, Guimbretière, and Hinckley	Papiercraft	Anoto	None	Active reading
2006	Chatti, Sodhi, Specht, Klamma, and Klemke	u-Annotate	Unknown	Unknown	Web browsing
2006	Plimmer and Mason	PenMarked	Tablet PC	None	Marking
2006	Plimmer, Grundy, Hosking, and Priest	RCA	Tablet PC	Content	Program code
2006	Priest and Plimmer	RCA	Tablet PC	Content	Program code
2006	Tront, Eligeti, and Prey	WriteOn	Tablet PC	Unknown	Lecture presentation
2006	Tront, Eligeti, and Prey	WriteOn	Tablet PC	Unknown	Lecture presentation
2006	Wang, Shilman, and Raghupathy	OneNote	Tablet PC	Unknown	Not specified
2007	Chen and Plimmer	CodeAnnotator	Tablet PC	Content	Program code
2007	Liao, Guimbretière, Anderson, Linnell, Prince, and Razmov	PaperCP	Multiple	None	Lecture presentation
2007	Plimmer and Apperley	PenMarked	Tablet PC	None	Marking
2007	Signer and Norrie	PaperPoint	Anoto	None	Lecture presentation
2007	Wang and Raghupathy	OneNote	Tablet PC	Content	Not specified
2008	Cattelan, Teixeira, Ribas, Munson, and Pimentel	Inkteractors	Tablet PC	None	Lecture presentation

Table 2: (continued).

Year	Authors	Implementation	Input Mechanism	Change Allowed	Application Domain
2008	Chang, Chen, Priest, and Plimmer	RCA & CodeAnnotator	Tablet PC	Content	Program code
2008	Liao, Guimbretière, Hinckley, and Hollan	Papiercraft	Anoto	None	Active reading
2008	Weibel, Ispas, Signer, and Norrie	PaperProof	Anoto	None	Editing documents
2008	Wu, Yang, and Su	Wu, Yang & Su (2008)	Unknown	Unknown	Collaboration
2009	Chandrasekar, Tront, and Prey	WriteOn	Tablet PC	Unknown	Lecture presentation
2009	Steimle	CoScribe	Anoto	None	Studying
2009	Steimle, Brdiczka, and Muhlhauser	Steimle (2009)	Anoto	None	Not specified
2010	Lichtschlag and Borchers	CodeGraffiti	Capacitive touch	Content	Program code
2010	Pearson and Buchanan	Pearson & Buchanan (2010)	Capacitive touch	Unknown	Collaboration
2010	Plimmer, Chang, Doshi, Laycock, and Seneviratne	iAnnotate	Tablet PC	Layout	Web browsing
2012	Chen, Guimbretière, and Sellen	United slates	eInk Reader	None	Active reading
2012	Hinckley, Bi, Pahud, and Buxton	GatherReader	Tablet PC	Unknown	Active reading
2012	Matulic and Norrie	Matulic & Norrie (2012)	Hybrid	Unknown	Active reading
2012	Steimle	CoScribe	Anoto	None	Collaboration
2013	Bhardwaj, Chaudhury, and Roy	Augmented Paper System	Visual	Unknown	Not specified
2013	Chen, Guimbretière, and Sellen	United slates	eInk Reader	None	Active reading
2013	Marinai	Marinai (2013)	Unknown	Layout	Not specified
2013	Mazzei, Blom, Gomez, and Dillenbourg	annOot	Tablet PC	Unknown	Studying
2013	Sutherland and Plimmer	vsInk	Tablet PC	Content	Program code
2013	Yoon, Chen, and Guimbretière	Yoon, Chen & Guimbretière (2013)	Tablet PC	None	Not specified

Table 2 lists the publications that were included in the mapping study, together with their high-level details. Full bibliographic information for all publications is in the reference list.

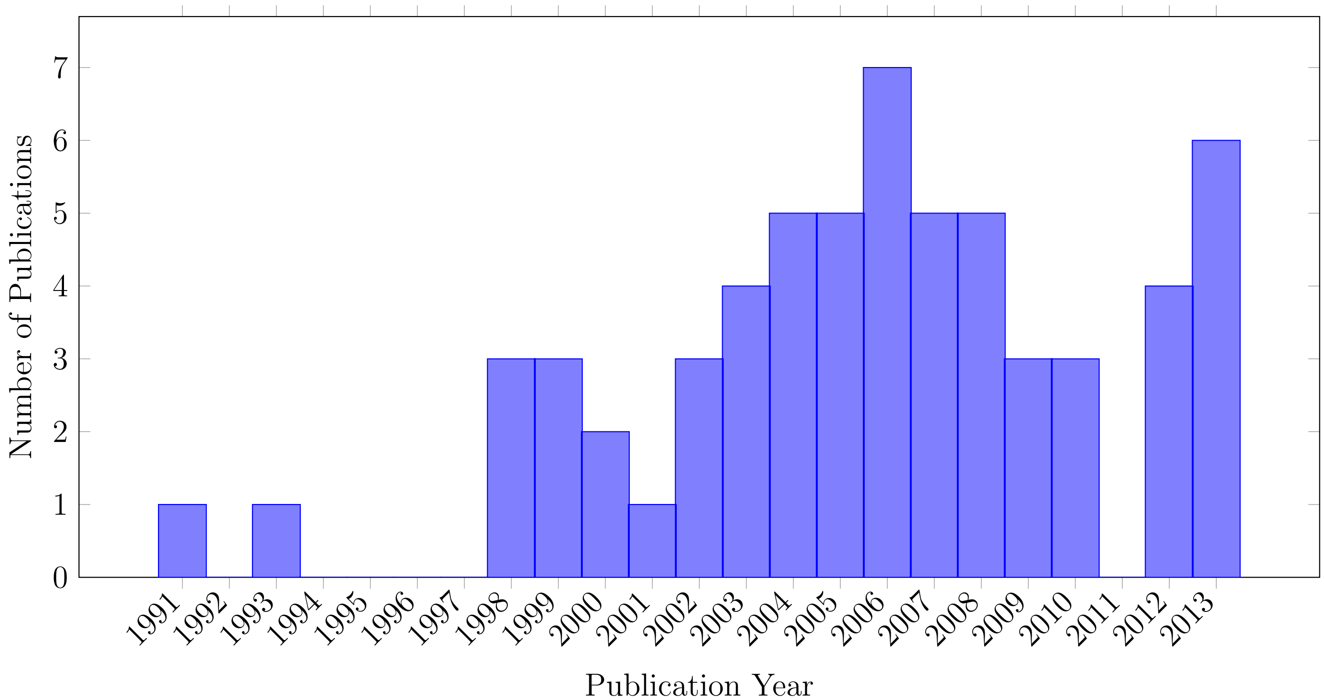


Figure 3: Number of publications included in corpus on freeform digital annotations from 1991 to 2013.

3.2. General Information

The earliest publication identified was published in 1991. Since then there have been between zero and five publications published a year (see Figure 3).

In this section we review the input mechanisms, change types supported, application domains and document formats.

3.2.1. Input mechanism

We were able to identify the input mechanism for 56 implementations (see Figure 4). We found the following input mechanisms: Tablet PC with stylus; Anoto pen; digitiser with stylus; PDA with stylus; customized eInk readers; capacitive touch and visual input.

The most common input mechanism reported is a stylus on a tablet PC (38 implementations). In this mechanism the user directly draws on the screen of the tablet PC with a special stylus. The tablet PC directly captures the digital ink which is then processed by the implementation. Eight

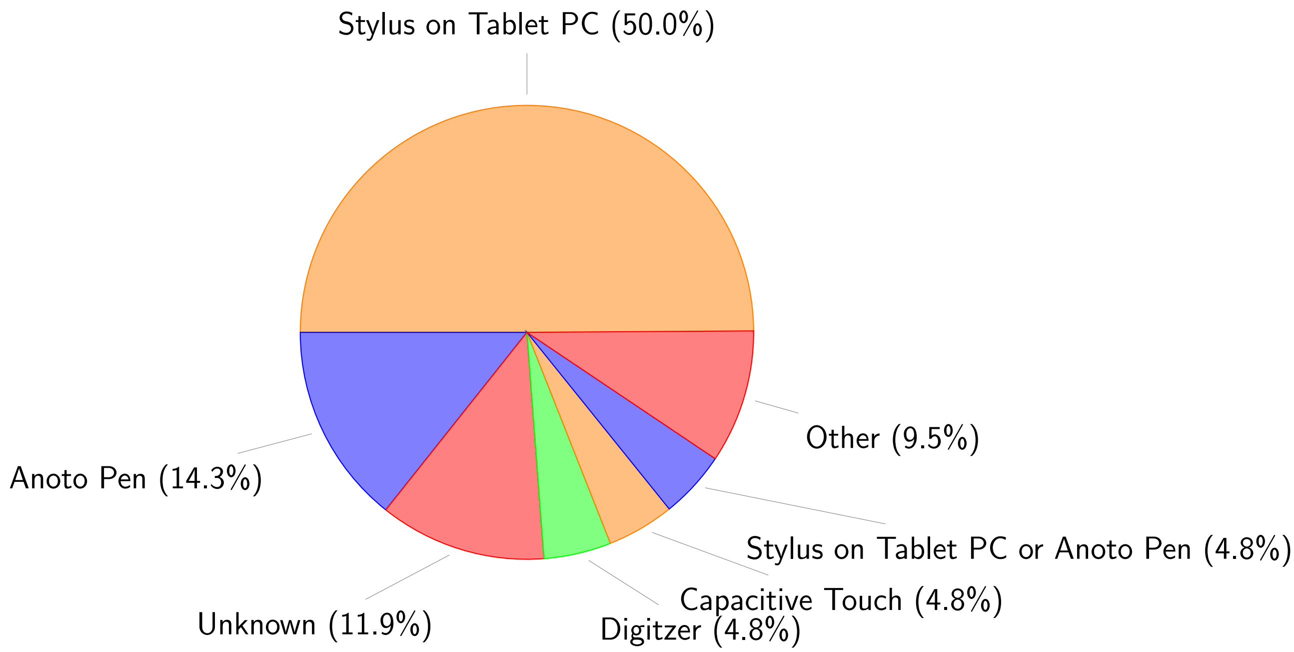


Figure 4: Percentage of implementations for each input mechanism.

implementations used an Anoto pen on paper. The document is printed and an Anoto pen records inking on the paper. The ink is then converted to a digital form, loaded into the implementation and added to the original digital document. A further two implementations allowed input from either the Anoto pen or a stylus on tablet PC. The remaining eight implementations used a variety of input mechanisms. Four used a stylus: two via a digitizer [37, 30], one with a PDA [44] and one used customised eInk readers [18, 19]. Two used touch on a capacitive touch surface [42, 56]. One used an Anoto Pen combined with a tabletop PC [50] and the final implementation used visual input to track the tip of a pen [9].

We identified three dimensions influenced by the input mechanism:

(i) directness;

(ii) accuracy;

(iii) physical size.

Directness describes the relationship between the input surface and the display. A direct mechanism (Tablet PCs and Anoto Pens) involves direct interaction with the display surface. In contrast, with an indirect mechanism (digitizer) there is a disconnect between the input surface and the document. The user has to map from the document to the input surface. With directness there are two possible interactions: input and output. Tablet PCs provide direct interaction for both input and output. The user can directly input ink onto the device and see it update. In contrast, Anoto Pens provide mixed directness: while the user can directly input on the display surface and see it update on the display surface, they do not directly see its digital representation. Digitizers also provide mixed directness: for input they are indirect but the user can directly see the digital representation. We did not identify any mechanisms that were indirect for both input and output.

Accuracy describes the level of precision when using the input mechanism. The most accurate input mechanism mentioned were the Anoto Pens. These have a theoretical precision of 0.03mm [50]. The least accurate input mechanism is using touch on a capacitive surface. Stylus input devices have a range of accuracies but few publications record any details on the level of precision achieved. However the precision will be lower than an Anoto due to the decreased display resolution compared to paper [3].

Physical size refers to the physical size of the input surface. The smallest device was the PDA for A-book [44]. The next larger devices are Tablet PC and eInk reader systems. Finally, the largest physical systems are the tabletop implementations. These systems require space for the table plus additional ancillary equipment (e.g. the camera in [9]). Anoto pen systems can potentially be anywhere on this dimension as the input surface depends on the size of the paper the document is printed on.

3.2.2. Change type allowed

Of the 61 implementations 3 supported layout-only changes, 11 supported layout-and-content changes and 35 did not support any type of change in the underlying document. For the remaining 12 we could not determine whether they supported any changes. The definitions of each change type are defined in §2.5.8.

3.2.3. Application domains

During the analysis we identified five main domains:

(i) Collaboration

(ii) General

  (1) Active reading

    (2) Document editing
    (3) Web browsing
(iii) Education;
    (1) Lecturing
    (2) Marking
    (3) Studying
(iv) Programming;
(v) Research.

Table 2 includes the application domain for each implementation.

Collaboration systems are primarily intended for communications between two or more people. In these systems annotations are a way of communicating information. For example, Wang Freestyle allowed people to exchange notes and documents. Annotations enhanced document exchange by including a simple way to add additional information [37].

General covers both reading and producing documents. The most common category in this document is reading documents: specifically active reading[2]. During active reading the reader uses a pen to mark the text as they read. XLibris, the most implementation with the most publications, was original designed as an active reading device [67]. Web browsing is another form of reading but with some key differences: active reading applications replicate how paper works [67, 68, 63] while web browsing focuses on the dynamic nature of web pages [17, 61]. Finally, document editing refers to the process of producing and editing documents. The three implementations in this category all focused on how an editor can annotate a document [30, 21, 84].

The most common area in education is lecturing: presenting slides to a class with annotation support [81, 5, 4, 33]. Some implementations investigated how annotations can help students when studying and taking notes [76, 51]. One implementation investigated how annotations improving marking of student work [59, 58].

The final two domains are subject specific and have a limited number of implementations. Some implementations looked at how annotations could be added to program code. These implementations mainly focused on the technical challenges of adding annotations within current IDEs and how they

[2]As mentioned by Chen et al. active reading also implies similar terms such as work-related reading and responsive reading [18].

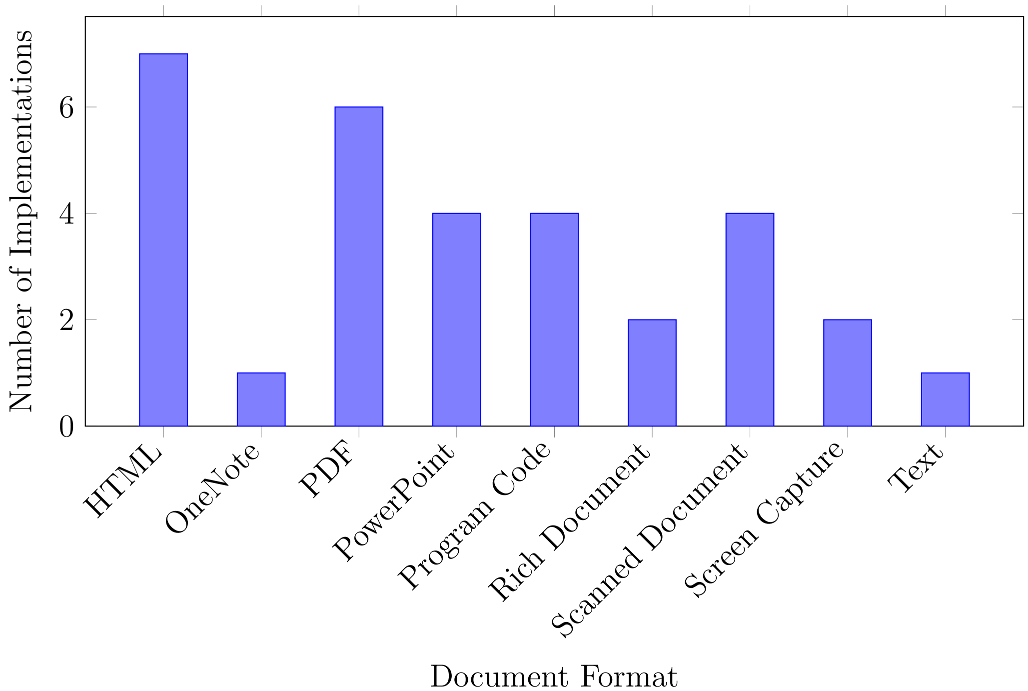


Figure 5: Number of implementations that handle each document format.

can assist with navigation [60, 64, 20]. However one also looked at how annotations could be useful for code in a collaborative environment [42]. The other domain was research: the single implementation in this domain looked at how annotations provide a link between physical and electronic documents [44].

3.2.4. Document formats

Many implementations do not mention document formats used (17 out of 42). The formats that are mentioned can be grouped into nine formats (see Figure 5). The most common format mentioned is HTML [8, 65, 17, 85, 74, 76, 61] (seven implementations) followed by PDF [29, 74, 76, 18, 19, 45, 86] (six implementations) and then program code [60, 64, 20, 16, 42, 77], PowerPoint [5, 72, 74, 76] and scanned documents [37, 62, 67, 68, 27, 48, 25, 26, 49, 24, 44, 71, 74] (all with four implementations). Scanned documents refers to documents that have been scanned and loaded into the implementation. Rich Documents refers to documents produced by word processing software: two implementations use this format (AbiWord [21]

Table 3: Taxonomy of Annotation Types

Type Catgeory	Examples
Single line	Underlines
	Highlighting
Multiple line	Enclosures
	Margin bars
	Braces
Connectors	Callouts/arrows
Complex	Text/symbols within text
	Drawings
	Marginalia
Commands	Commands

Table 4: Taxonomy of Annotation Support Operations

Category	Operation
Adding operations	Grouping
	Recognising
	Anchoring
	Storing
Adapting operations	Repositioning
	Refitting
	Orphaning/Deleting

and OpenOffice [84]). Screen capture refers to a direct capture of the screen: two implementations use this format [15, 79, 80, 55].

3.3. Taxonomy

A summary of the research approaches to digital ink software is presented in Tables 3 and 4. This taxonomy has been compiled as a result of the data synthesis step (see §2.6). Table 3 lists the categories of annotation types and examples of each type. Table 4 lists the adding and adapting operations.

3.3.1. Annotation types recognised

It is clear from the various studies that been conducted (e.g. [46, 24, 47, 8, 70, 83, 78]) that there are some common types of annotations. However

Table 5: Annotation category recognised by implementation

Implementation	Single Line	Multiple Line	Complex	Connector	Command
Callisto [8]	Y	Y	Y		
Classroom Presenter [5, 6, 4]	Y	Y			
CodeAnnotator [20, 16]		Y		Y	
Intelligent pen [28]	Y	Y	Y		Y
MATE [30]					Y
Matulic & Norrie (2012) [50]					Y
OneNote [83, 82]	Y	Y	Y	Y	
PaperCP [39]					Y
PaperPoint [72]					Y
PaperProof [84]					Y
Papiercraft [40, 41]					Y
ProofRite [21]			Y		Y
Ramachandran & Kashi (2003) [65]					Y
RCA [60, 64, 16]		Y		Y	
ScreenCrayons [55]	Y	Y			
Shilman & Wei (2004) [70]	Y	Y	Y	Y	
Steimle (2009) [74]					Y
United slates [18, 19]			Y		
vsInk [77]	Y	Y		Y	
Wu, Yang & Su (2008) [85]	Y	Y	Y		
Xlibris [62, 67, 68, 27, 48, 25, 26, 49, 24, 71]	Y	Y	Y	Y	Y
Yoon, Chen & Guimbretière (2013) [86]					Y
Number of Implementations:	9	11	8	6	12

these are dependent on both the individual and the domain. Despite this limitation many implementations still attempt to recognise the annotation type as this allows for additional functionality later.

The annotation types recognised could be determined for 22 implementations. The remainder of the implementations either do not handle specific annotation types or do not describe the annotation types recognised. Table 5 lists these implementations and the categories that they recognise.

Ten different annotation types were found. The following are the definitions of each annotation type:

(i) An underline is a line drawn underneath or through a sentence.
(ii) A highlight is similar to an underline but drawn with a different (often semi-transparent) pen.
(iii) An enclosure is a border around one or more elements.
(iv) A margin bar is a vertical straight line drawn in a margin.
(v) A brace is similar to a margin bar but has a pronounced rounded shape and a centre prominence.
(vi) An arrow or call out is a line drawn from one element to another. It may have arrow heads on one or both end points.
(vii) Text and symbols are characters written in the body of the underlying text. They are generally added in the whitespace around the underlying text.
(viii) A drawing is a picture or diagram.
(ix) Marginalia are longer notes added in the margin.
(x) Commands are marks that the implementation is expected to understand and execute.

Figure 6 shows the breakdown of which annotation types are commonly supported.

Different annotation types have different requirements from a software perspective. These annotation types were grouped into five categories based on the requirements of each type:

(i) Single line (underline and highlighting): these annotations are associated with a single line in the document;
(ii) Multiple line (enclosures, margin bars and braces): these annotations span multiple lines;
(iii) Connectors (arrows/callouts): these annotations associate two areas or annotations together;

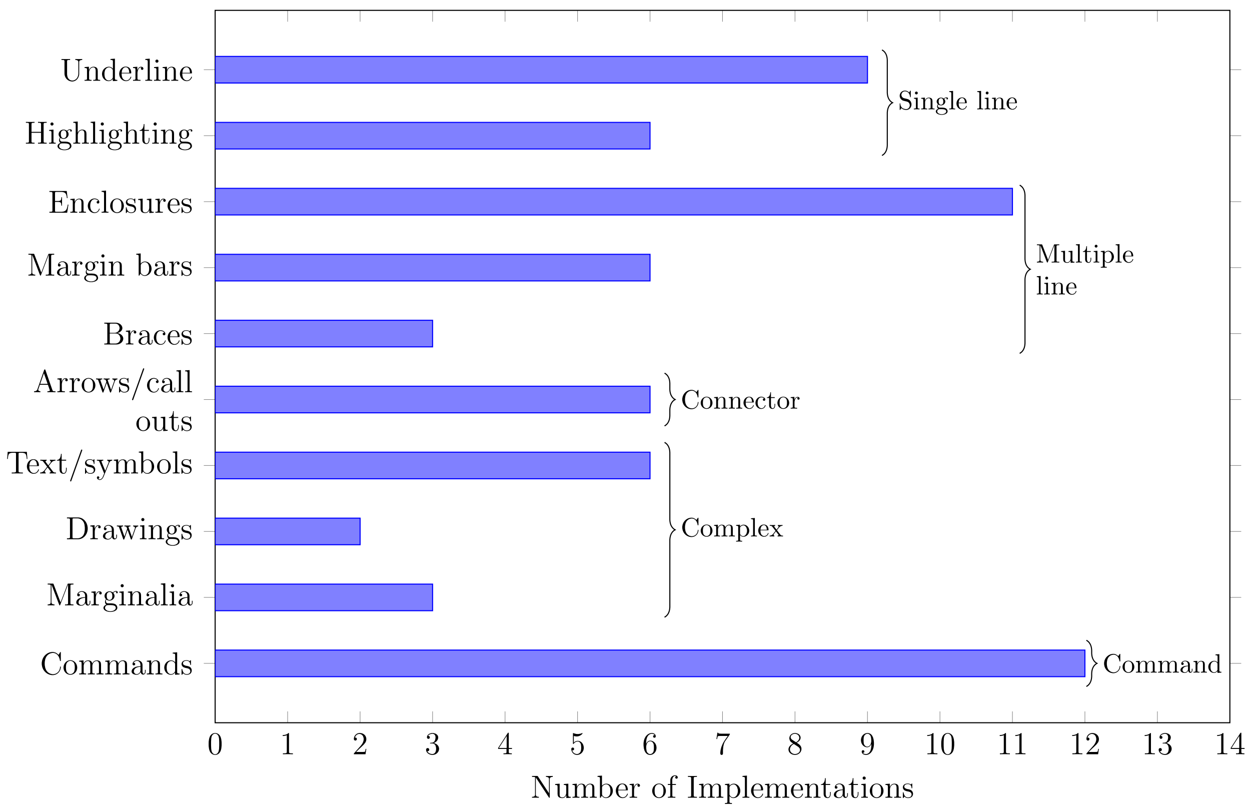


Figure 6: Number of implementations that recognise each category of annotations.

(iv) Complex (text/symbols, drawings and marginalia): these annotations have additional meaning in addition to their location.

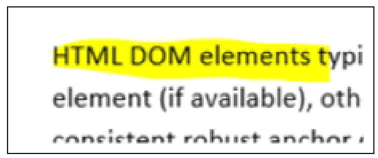
(v) Commands (commands): these marks are commands for the system to perform. These are usually a limited set of symbols that the system can recognise.

Table 6 shows the five categories and an example of each of the types.

Table 6: Examples of each annotation type.

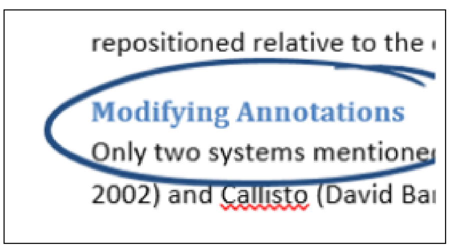
Single Line Annotations

Underline

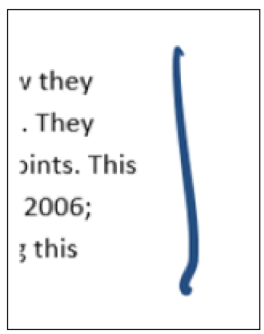


Highlight

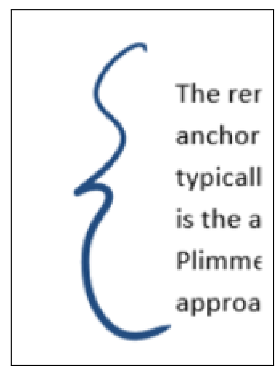
Multiple Line Annotations



Enclosure

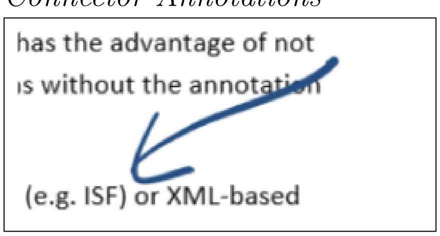


Margin Bar



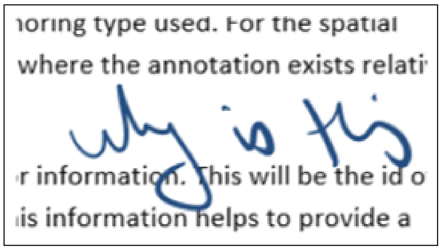
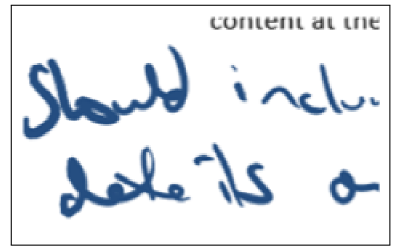
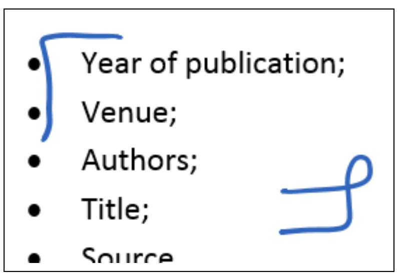
Brace

Connector Annotations



Callout/arrow

Table 6: Continued.

Complex Annotations		
		
Text/symbol	Drawing	Marginalia
Commands		
		
Command		

Single line annotations were recognised by nine implementations. All nine implementations recognised underlines; six also recognised highlights.

Multiple line annotations were recognised by eleven implementations. All eleven implementations recognised enclosures; six recognised margin bars and three recognised braces.

Connectors were recognised by eight implementations.

Complex annotations are recognised by six implementations. All six implementations specially mentioned they recognised text/ symbols; two recognised drawings and three recognised marginalia. These three implementations mentioned marginalia as a separate type; other implementations may have supported the same type but reported them as text/symbols.

Finally, commands were recognised by twelve implementations.

In addition to these categories, annotations fit into two classes based on their intended use. The first class of annotations are those intended for a person. While a computer may recognise these annotations they often have additional meaning beyond their appearance and location. The second class of annotations are those intended for the computer. These annotations can be completely understood by the computer. All commands fall into this category as the computer must understand them in order to apply them.

One challenge with annotation systems is dividing annotations into the two classes. If the annotation is intended for the application there needs to be some way of recognising these annotations; otherwise the application will treat them as intended for a human. The publications reviewed in this study describe the following approaches:

(i) Pen buttons;
(ii) Separate display space;
(iii) Special gestures;
(iv) Pen and touch.

Pen buttons involve using one or more buttons on the stylus device. When the user wants to change modes they depress these buttons. The buttons can either change the mode until the button is pressed again or only change the mode while the button is depressed. With the second option the mode returns to the original when the button is released.

With a separate display space there is an area of the screen where gestures must be entered. Any ink outside this area is assumed to be human-readable only. Gestures within this area will be recognised and potentially processed.

Special gestures involve either a specific gesture set or a special gesture that is added to other gestures. With the specific gesture set the implementation attempts to recognise all gestures. If the gesture is recognised then it will be processed; otherwise the implementation assumes that it is human-readable ink instead. With the special gesture the implementation will ignore all ink unless it includes the gesture. If the special gesture is included then the entire gesture is assumed to be a command.

Finally, with pen and touch one hand is used to control the pen and another to provide touch input to the implementation. Based on the touch input th the implementation either treats the pen input as human readable or as commands.

3.3.2. Adding operations

Adding an annotation to a document involves several steps. Digital ink is captured as ink strokes. A single ink stroke is generated by a pen-down,

Table 7: Adding and Adapting operations by implementation

Implementation	Group	Recognise	Anchor	Store	Reposition	Refit	Orphan
Callisto [8]	Y	Y	Y		Y	Y	
Classroom Presenter [5, 6, 4]		Y					
CodeAnnotator [20, 16]	Y	Y	Y		Y		Y
CodeGraffiti [42]			Y		Y		Y
CoScribe [76, 75]		Y	Y				
iAnnotate [61]	Y		Y	Y	Y		
Intelligent pen [28]		Y		Y			
Marinai (2013) [45]					Y		
MATE [30]		Y					
Matulic & Norrie (2012) [50]		Y					
OneNote [83, 82]	Y	Y	Y				
PADD [29]			Y	Y			
PaperProof [84]		Y		Y			
Papiercraft [40, 41]		Y					
Pearson & Buchanan (2010) [56]				Y			
ProofRite [21]			Y	Y	Y	Y	
Ramachandran & Kashi (2003) [65]			Y	Y		Y	
RCA [60, 64, 16]	Y	Y	Y	Y	Y		Y
ScreenCrayons [55]		Y					
Shilman & Wei (2004) [70]		Y	Y				
Steimle (2009) [74]		Y					
u-Annotate [17]			Y	Y			
United slates [18, 19]			Y	Y			
vsInk [77]	Y	Y	Y	Y	Y		Y
Wu, Yang & Su (2008) [85]		Y	Y	Y			
Xlibris [62, 67, 68, 27, 48, 25, 26, 49, 24, 71]	Y	Y	Y	Y	Y	Y	
Number of Implementations:	7	17	16	13	9	4	4

pen moves, pen-up sequence. However people do not consider ink strokes individually. Instead they cognitively group them together as annotations. An annotation can consist of a single ink stroke (e.g. an underline or enclosure) or multiple strokes (e.g. text or drawings).

We found four operations involved in adding an annotation to a document:

(i) Grouping: combining multiple ink strokes into a single annotation;
(ii) Recognition: classifying all or part of the annotation;
(iii) Anchoring: determining the location of the annotation relative to the underlying context (further details are mentioned in §3.3.3);
(iv) Storage: persisting the annotation details, including information about the anchor;

Very few publications reported on how annotations are stored. Most implementations that store the annotations use a separate file for the annotations (i.e. they do not modify the original document). These files are either stored locally on the user's machine [17, 64, 77] or sent to a server [61]. Another approach is to modify the original file to store the annotations in it [18, 83, 82].

Binary and XML-based formats are the only two storage formats mentioned. Only one publication describes the storage format in detail [65] although some other implementations do use pre-defined formats (e.g. Microsoft's ISF format) [64, 77].

Different implementations use different sequences of these steps. The sequence of steps implemented depends on the goals of the implementation. However there are two general sequences: all-annotation and single-annotation.

The all-annotation sequence processes all strokes in the document when a new stroke is added. The first step is to group strokes into annotations. Then the implementation attempts to recognise each annotation. The result from the recognition (the annotation type) is used to anchor the annotation in the document. Finally the annotation details are stored. With this sequence there is more information for the grouping and recognition steps. This may improve recognition accuracy but comes at the cost of increased computation. This increase is due to the need to reprocess all strokes. The implementations by Shilman and Wei [70], Wang et al. [83], Wang and Raghupathy [82] are examples of this sequence.

The single-annotation sequence processes each stroke only once. When a stroke is added the first step is to group the stroke with an existing annotation. If this is not possible then a new annotation is started. The implementation recognises the annotation and anchors it to the document. Recognition and anchoring are only applied to new annotations. No matter how many strokes are added to an existing annotation the type and anchor do not change. Finally the annotation is stored. This sequence requires less rework as recognition and anchoring is only performed once per annotation but this may reduce recognition and anchoring precision.

The single-annotation sequence can involve the user in the grouping operation. The amount of user involvement ranges from the user manually doing all the grouping to the implementation providing hints. Callisto requires the user to do the grouping (and recognition) [8]. RCA, CodeAnnotator and vsInk all provide grouping hints by displaying a border around the annotations. When the new stroke is inside or intersects an annotation's border it is included with the annotation [20, 64, 77]. XLibris does not provide any user feedback or involvement in the grouping [24]. Instead it uses timing and spatial heuristics to automatically group strokes together.

Not all implementations use all four operations. Some implementations do not mention any form of recognition (e.g. [17, 61]). Other implementations treat all strokes as individual annotations and do not mention any grouping (e.g. [70, 85]).

There are also notable exceptions to the overall sequences listed above. These typically include one or more of the steps but don't do it for the purpose of adding annotations. For example recognition is used to separate temporal, attention strokes from permanent ink annotations [6]; to identify sections in a document that would be most useful to the user [71]; and to apply a mask to the document to emphasize what was annotated [55].

There have not been any comparative studies between these sequences to determine the relative efficacy of each.

3.3.3. Anchoring mode

Annotation anchoring involves associating an annotation with an element of context in the document. All current research treats freeform ink annotations as graphical elements. These graphical elements are positioned in the document relative to a bounding box. We have classified the anchor mode by the type of bounding box used (see Table 8).

The most common approach is to use the whole page as the bounding box. Both the document and the annotations are treated as graphical representations. Typically the annotation's top left corner is recorded as an offset

Table 8: Number of implementations for each anchoring approach

Anchoring Approach	Number of Implementations
Whole page	20
HTML Element	4
Code Line	4
Unknown	10
Word	2
Paragraph	1
Total:	41

from the top-left of the page. The annotations are merged onto the document to produce the final view. While this approach is simple and easy to implement it does not allow for the underlying document to change. If the document changes a new graphical representation needs to be generated and the associated coordinates either lose their meaning or need to be translated. There are no reports of translating an annotation to new coordinates without using a more sophisticated approach to anchoring.

The remaining approaches all use a smaller element on the page. The page is decomposed into these elements and the closest element is selected as the bounding box. Where there are multiple choices the implementation will use some form of preferential ordering to select the "best" bounding box [82, 77].

Both paragraph and word approaches use the words in the document as the anchor. The anchor can include using the words themselves, using a number to identify the word within the document (e.g. words 10 to 15) or the location of the words (e.g. words 1 to 5 in the second paragraph of the third page) [24]. All of these approaches assume the words do not change within the document. This approach does however support reflow of the existing text, for example the font size being changed so that words flow onto other lines [24].

Anchoring with an HTML element uses the underlying HTML document object model (DOM). HTML uses a tree-like structure for generating a page. The browser renders this structure into a graphical representation that the user sees. Choosing an anchor position involves finding the closest element to the annotation. The information stored for the annotation includes the

Table 9: Number of implementations that automatically adapting annotations

	Repositioning	Refitting	Orphaning
Layout-only	3	2	0
Layout-and-content	6	2	4
Total	9	4	4

identifier of the element (if any), the path from the root to the element and the surrounding elements [17, 61].

The bounding box for a code line is around each individual line. The anchor for code annotations consist of the line number and file name [20, 64, 77].

3.3.4. Adapting operations

When the underlying document changes an annotation may need to adapt in response. By adapting the annotation it retains its meaning and value. The actual type of adaptation depends on how the underlying document is modified.

There is less work published on automatically adapting annotations. Only ten out of the 31 implementations mention any form of automatic adaptation, we grouped these into three categories:

(i) Repositioning: the annotation is moved to a new location;
(ii) Refitting: the appearance of the annotation is changed;
(iii) Orphaning: the underlying context for the annotation has been removed.

In addition to these categories, implementations can be classified by the type of document modification that is handled. Table 9 shows the relationship between these two categories.

Nine implementations handle repositioning annotations when the underlying content changes, four them handle orphaning and four refit annotations. There was only one implementation that handled refitting but not repositioning [65]. This is unusual as normally repositioning is easier to implement than refitting. It may be this implementation does handle repositioning but it was not mentioned in the publication. All systems that implement orphaning also implement repositioning.

Four out of the six implementations that handle content changes are code editors. The others are ProofRite and Callisto [21, 8]. All four implementations that handle orphaning are also code editors. They all use the same approach for handling orphaning by deleting the annotation. None of the publications on these implementations mention anyway for the user to review the orphaned annotations [20, 42, 60, 64, 77].

The effectiveness of annotation repositioning is related to anchoring. Repositioning requires an anchoring mode at a more granular level than whole-page. All nine implementations that support repositioning use a more granular mode. Four implementations used a line bounding box [64, 20, 42, 77]; one used a bounding box based on HTML elements [61]; one used paragraph level bounding boxes [45]; two used word level bounding boxes [24, 8]; the final implementation did not mention how the annotations were anchored to the context [21].

Repositioning calculates the new position of the annotation using the position of the anchor element plus an offset. The first step is to retrieve the current location of the reference point. The offset is then added to this reference point and the annotation positioned using the sum. All approaches are in reality using (x, y) co-ordinates (as the annotation is a graphical element) translated relative to the anchor [77].

Only four implementations mentioned any refitting of annotations: XLibris [24]; Callisto [8]; ProofRite [21] and the system by Ramachandran and Kashi [65]. All of these implementations use similar rules. Two systems only handle layout changes (XLibris and Ramachandran and Kashi). For XLibris the rationale was to remove any confounding influence due to not finding an anchor for an annotation. Callisto and ProofRite handle both content and layout changes.

In XLibris single-line annotations (underlines and highlighting) remain attached to the words they are anchored to. If a line splits then the annotation will also split; if two lines with similar annotations are joined the annotations will also be joined. Multi-line annotations (enclosures and margin bars) are stretched or condensed so the top and bottom margins of the annotation stay in the same relative positions to the underlying context. Complex annotations are not refitted [24].

Callisto treats enclosures as a single line annotation and associates them with the line in the same way XLibris handles underlines and highlights. Braces are handled instead of margin bars. Callisto also has a mode where the annotations are converted to "cleaned" annotations. Underlines and highlights are converted to straight lines that align with the underlying text; enclosures are converted to rectangles with rounded corners and aligned to the underlying text; braces are converted to simple Bezier curves. Once cleaned the annotation then follows the same refit rules as the original annotations. The rationale for this is it is easier to automatically refit "cleaned" annotations [8].

ProofRite follows the same rules as XLibris [8]. The system by Ramachandran and Kashi does not describe the rules for adapting annotations [65].

Table 5 shows which implementations recognise the different categories of annotations. Table 7 shows the implementations which implement adding and adapting operations.

3.4. Additional Functionality

While the main focus of this review is how to add and adapt freeform ink annotations on digital documents we also recorded additional functionality that is possible in a digital environment. We report this functionality in this section for completeness. However due to the wide range of functionality that is available we do not cover them in detail.

Table 10 outlines additional functionality provided by the different implementations. This functionality extends annotations beyond what is available using pure pen and paper. During the data synthesis phase (see §2.6) these were grouped into seven categories: collaboration; distributed; dynamic support; intelligent support; navigation; viewing and other. These are described below.

Collaboration is about sharing annotations between people. At its simplest this involves sharing annotations one person makes with another. There are variations on this simple theme. First, annotations can be shared equally between different people (Sharing Annotations). In this variation each person is treated as an equal collaborator (at least at the implementation level). In contrast *interactive lectures* implies unequal sharing: one person is giving the lecture and the rest are students participating. Interactive lectures refers to either sharing the lecturers annotations to a wider group [5, 6, 4, 39] or the lecturer viewing students' annotations [81, 39]. Another issue that is specific to sharing annotations is privacy . People often make annotations that they consider private and they do not want to share with other people [13, 81, 47, 75]. Thus some implementation have looked at how the privacy settings can be changed [81, 75].

Table 10: Additional functionality provided by computer-assisted annotations

Category	Functionality	References
Collaboration	Annotation privacy	[75, 81]
	Interactive lectures	[5, 22, 39, 81, 6, 4, 72]
	Sharing annotations	[37, 33, 42, 51, 56, 75, 74, 85]
Distributed	Control of remote device	[72, 4, 39, 18, 19]
	Shared information between devices	[18, 19]
Dynamic support	Automatic adaptation	[8, 20, 24]
	Manual changes	[64, 77, 20]
Intelligent support	Automatic word lookup	[9]
	Command execution	[40, 41, 50, 65, 72, 84, 21, 30, 84]
	Error recognition	[5]
	Gesture recognition	[28, 40, 41, 50, 65, 72, 84, 21, 30, 84]
	Handwriting recognition	[6]
	Masking	[55]
Navigation	Automatic collection of annotations	[49, 62, 67, 63]
	Automatic list of related materials	[63, 67, 62]
	Automatic search based on annotations	[27, 62, 67, 71, 68]
	Hyperlinking	[9, 25, 67, 62]
	Linking documents	[41, 74, 76]
	Navigation based on annotations	[20, 18, 30, 67, 77]
	Tagging documents	[74, 76]
Viewing	Additional annotation space	[4, 86]
	Replay of annotations over time	[14]
	Zooming	[3]
Other	Combining pen with touch	[50]
	Direct screen capture and annotation	[15, 80]
	Information gathering	[31]
	Supporting workflow	[59, 58, 57]

Some implementations offer distributed functionality: the system is separated across multiple machines. There are two variations: controlling a device or sharing information. Controlling a device assumes there is master device and one or more devices being remotely. For example, lecture presentation systems often allow the lecturer to control the presentation without being anchored to the device that controls the projector [72, 4, 39]. In contrast, sharing information is where all devices are considered equal [18, 19]. Because the information is shared the user does not have to worry about what device they entered the data on. This overcomes some of the earlier limitations found when comparing pen-and-paper to electronic annotation making [53, 52].

Functionality in the dynamic support category is about changing annotations after they have been added. This category divides into two groups: automatic adaptation and manual changes. With automatic adaptation the implementation tries to change the annotation in a way that the original meaning is preserved [8, 20, 24]. The user is either not or minimally involved in the changes. In contrast, manual changes is where the user has full control over how the annotations are changed [64, 77, 20]. This can be changing the location of the annotation, completely deleting it or changing how the annotation's appearance.

Intelligent support adds contextually aware functionality to annotations. The majority of the research in this category requires recognition (e.g. recognising errors, functional gestures, and hand-writing). The premise behind recognition is if the system can understand an item it can then add extra support for it. For example, if a gesture can be recognised then an associated command can be executed automatically. Thus command execution is flow-on functionality from recognition. Other intelligent functionality includes looking up words and masking out parts of the document based on the underlying context of an annotation [9, 55]. Both of these functions aim to reduce the workload on the user by anticipating their intentions.

Navigation is another common category. For example, XLibris explored many different ways of navigating through documents based on a user's annotations [62, 63, 67, 68, 27, 49]. These include collecting annotations together so the user can view them in one location; using hyperlinks to return to the original source; and generating lists based on the context of the annotations [62, 63, 67, 68, 27, 49]. Other navigation functionality includes the user linking documents together via annotations or tagging them for future reference [41, 74, 76].

The next category, viewing, is how to display annotations in different ways. One commonly reported issue with annotations is the amount of space available for them. Some implementations have looked at how to increase space automatically without user interventions [4, 86]. Another approach is to replay annotations in chronological order so the viewer can see how they have been built up [14]. The third approach is to help with inputing annotations (in order to overcome of the limitations in technology) [3].

Finally, remaining niche functionalities are grouped together under 'other'. For example, combining pen and touch interactions together [50]; using screen capture to generate documents [15, 80]; information gathering [31] and supporting workflow [59, 58, 57].

3.5. User Studies

We found a variety of user study types reported. We divided them into three types:

(i) Usability;
(ii) Technical capacities;
(iii) User expectations.

Usability studies investigate the effectiveness and efficiency of the implementation. There are a number of usability studies that looked at the usefulness and learnability of various implementations [4, 24, 45, 51, 75]. These studies have identified a range of issues that need to be considered. However few of these studies took their results and generalised them beyond the implementation. This makes these results specific to the implementation itself without exploring the wider possibilities of what it means for user expectation.

Studies on the technical capacities investigate the technical limitations of the implementation. These might include speed, performance, and accuracy. Again these studies are limited to the implementation; although the details do often show where the implementation can be improved.

The final type of study is on user expectations. These investigate new avenues that are not possible with paper-based annotations. XLibris is an example of an implementation that was used to investigate user expectations in a variety of contexts [62, 67, 27, 48, 49, 71]. While these studies are interesting and provide more detail on user expectations for this review our specific focus on user expectations is around the automatic adaptation of annotations. There is only one study in this area [8].

One important finding is that users like the implementations that are predictable and reliable. However if this is not possible then they do not want the implementation to change their annotations. Bargeron and Moscovich [8] found users would prefer the underlying text to be locked, so they cannot modify it, if the annotation cannot be accurately adapted. They theorised there would be a cut-over point for when to lock the context but they were unable to detect one based on their results.

Another important finding is people are happy with "cleaned" annotations. These annotations are often preferred over the original annotations. In addition people are happier seeing these annotations change in response to changes in the underlying document than seeing their original annotations change. However cleaning the annotations increases user expectations. The users have a higher expectation that the implementation understands their meaning [8].

While only one study specifically looked at user expectations around adaptation there are several other studies that include results related to automatic adaptation. One area where annotations do not always behave as expected occurs in the grouping operation. Some annotations (e.g. text, drawings, etc.) are expected to remain together. For example the cross stroke of the 't' and the dot above the 'i' should remain in position relative to the rest of the letter. In some automatic implementations this does not happen during repositioning [24, 45].

Another area that can cause confusion is resizing multiple line annotations [8, 24]. When the annotation is outside the text this is not an issue (e.g. margin bars or braces) but when the annotation is within the text the meaning is not preserved as effectively. One potential reason for this is adaptations for this category of annotation do not take into account which words the annotation should be associated with.

Anderson et al. [4] suggest that digital annotation can be more difficult to read than annotations on paper. Identified factors that cause this include:

(i) Pen size: often the pen is a larger size than would be used on paper. The annotations can take up too much space on the document and obscure the underlying text;

(ii) Pen colour: the colours chosen for the annotations can make them more difficult to read (especially when displayed via a projector);

(iii) Annotation similarity: all annotations added using digital ink have the same colour. Unless the user changes the colour all ink annotations in the same location will merge together.

These factors make it harder for people to accurately add annotations to the document. This reduction in accuracy then has a flow-on effect where annotations are more difficult to correctly adapt. DIZI is an example of a system that attempts to overcome these challenges [3].

Anderson et al. [4] further suggest the last point occurs because digital ink doesn't change over time like pen annotations do. They claim when ink is first added to paper it appears slightly different. Then as time passes the ink dries and the colour changes slightly. This makes it easier to differentiate annotations based on the time they were added. The colour also changes when multiple strokes are layered on top of each other with a pen. But with digital ink all the strokes have exactly the same colour.

4. Discussion

During the review we found a number of areas of significance. In this section we discuss these and how they impact on freeform digital annotations. In §4.1 we address one of the challenges in this review, the plethora of terms used, by providing a set of common terms. In §4.2 we discuss how the various input mechanisms influence the functionality available. In this review we identified three adapting operations reported: repositioning, refitting and orphaning. Before these operations can be applied there are four adding operations that influence automatic adaptation: grouping, recognising, anchoring and storing. The effectiveness of the adapting operations depends on the effectiveness of these adding operations. Therefore §4.3 discusses the adding operations followed by §4.4 on the adapting operations. One important factor that influences adaptation is the type of the annotation. Previous work has identified different types of annotations but these are normally limited to static documents. In §4.5 we describe how our taxonomy handles dynamic document and use this to identify gaps in the current literature. Another important concept that influences adaptation is annotation lifetime; in §4.6 we discuss this and how it interacts with fluidity. In §4.7 explain why other functionality warrants a new review. Finally, in §4.8 we address a gap in the literature around user experience and going from individual usability studies to the wider picture.

4.1. Terms

One of the issues we found during this review is different publications use different terms for the same thing. We propose the following set of terms being either the most common term used or that which most clearly describes the feature.

(i) Gesture: a pen or touch stroke from contact point through movement on the surface to lift off.
(ii) Digital ink: gestures that remain on the surface as visible ink.
(iii) Functional commands: gestures that result in a command (e.g. undo).
(iv) Annotation: a group of logically related digital ink gestures.
(v) Markup: a general term for all the annotations on a document.
(vi) Lifetime: the expected duration of an annotations existence.
(vii) Fluidity: the interaction experience of the user; in particular whether mode changes are necessary.
(viii) Adaptation: the alteration of annotation so that they continue to hold their meaning when the underlying document changes.

4.2. Input mechanisms

The *input mechanisms* for annotations affect the functionality that software can provide. Based on the literature we define three dimensions: directness, accuracy and physical size. Choosing an *input mechanism* involves a trade-off along these dimensions. For example, An Anoto pen allows the user to retain directness of input and immediate output, plus high accuracy, but at the cost of losing the directness of digital output. Thus the user knows what is happening on the paper but not on the computer; and should the digital representation change the user would be unaware unless they notified. In contrast, a Tablet PC is very direct for both input and output: the user is fully aware of what is happening at all times. The price for this trade-off is the reduction in accuracy.

In addition to these dimensions there are other factors that influence the choice of *input mechanism* that are not captured in these three dimensions. These factors are harder to classify on a scale but still have an influence on the user experience. For example, Anoto pens use real paper as the input and display surface. This provides the full range of affordances that are available for paper (e.g. physicality, freedom of interactions, ability to spread out, etc.) but also paper's limitations (e.g. being static, taking more space, unable to search, etc.) In contrast, tablets are full computers with all the processing ability that computers have. Users can tap into this functionality as part of their experience; but at the cost of a bulkier device and losing some of the affordances of paper.

The majority of the implementations reported on use a tablet device for the *input mechanism*; with pen-and-paper UIs (Anoto pens) being the second most common *input mechanism* (see Figure 4). From a research perspective the main differential is whether the annotation is on paper vs. on screen with the occasional attempt to combine the two modalities (e.g. [50]). We theorise that tablets are most popular because they are fully functional computers with the added input modality of ink. Thus the research in this area is around how to use digital ink as the input modality and what benefits it provides compared to other input modalities. In contrast, using pen-and-paper UIs are approaching digital ink from the viewpoint of how can paper be extended with a focus on two areas. The first area is what additional functionality can be provided beyond a pure pen and paper environment (e.g. collaboration [76, 75] and lecture presentation [39, 72]). The second area is how to overcome the limitations of paper (e.g. lack of feedback [50] and lack of interactivity [74, 84, 41]). Of interest, initial research only used tablet PCs but now pen-and-paper UIs are gaining in popularity. This may be due to initial hardware limitations being overcome or the realisation that paper still offers many affordances that computers have yet to replicate. Overtime it may be possible for these two approaches to come together with their synergies feeding off each other. However this combination of approaches appears to be limited by the current hardware available.

With the current state of technology, it appears that most of the technical annotation issues for static documents have been at least partially solved. This is especially true for annotating using pen-and-paper UIs but also for tablet-based implementations. What remains for static documents is the process of improving what is already available. However there are increasing numbers of devices with mobile touch/pen input which suggests that it is timely to consider dynamic documents more. This is especially relevant around how to handle changes when the underlying document context changes.

4.3. Adding operations

Grouping is the process of combining multiple gestures together into a single annotation. This is required because people do not think of annotations as individual gestures but a whole while computers receive the digital ink as individual gestures. When the gestures are grouped together successfully then the entire annotations appears to be one whole unit. One common problem with adapting an annotation is when individual parts of the annotation move independently of the others [24, 45]. This then causes confusion as the user expected the whole to stay together.

Recognition is the process of understanding either part or all of the annotation. This is important because different annotation types require different actions [24, 8]. For example, adapting an underline requires the underline stay underneath the associated words. If the annotation is incorrectly recognised then the wrong action will be applied to it, again resulting in user confusion. Another potential area of investigation that relies on recognition is cleaning annotations [8]. Cleaning annotations potentially provides an intermediate representation that is easy for processing but still remains understandable to the user. However they must be correctly recognised to remain understable; incorrect recognition would result in a wrong cleaning operation being applied.

Anchoring is the processing of associating the annotation with a location in the underlying document. Anchoring is a key prerequisite for repositioning: the annotation will only move to the correct location if the anchor is correct. However, as identified in the literature, there are a variety of different anchoring approaches. The simplest approach is to associate the annotation within the page. This only requires a graphical offset to the top of the page without any need to identify individual elements on the page. But what this gains in simplicity it loses in functionality: there is no way to do any adapting operations at a lower level than the page. In contrast, the most granular level is to identify individual words (or even letters). But this approach raises more issues: first, how to correctly identify the anchor word; second, how to find this word again after the document has changed; and third, how to adapt the annotation if it spans multiple words. Other approaches have used less granular anchors (line or paragraph level) or anchors based on the underlying document code (e.g. HTML). What is common about all these approaches is there is a trade-off. Moving to a more granular level allows more control and flexibility but at an increasing cost of complexity. As yet, there is no clearly identified "best" approach. Instead it depends on what the implementation is trying to do and how important it is to accurately adapt the annotation.

The final operation, storage, is less important for adapting annotations. Its impact is around how annotations can be retrieved later on. For implementations that are only interested in immediately evaluating the adding or adapting operations, this operation can be omitted altogether. However for implementations that need to be persisted this operation is important. One current issue with this operation is the data that is stored. There is not a common data format or storage location: thus each implementation needs to implement this operation itself. There are some common formats (e.g. InkML or Microsoft's ISF format) but these appear to be focused at the digital ink level rather than the higher level of annotations. Thus using these formats requires extensions to include relevant information.

Both anchoring and storing have well-studied solutions. There are sound solutions for anchoring that work in most circumstance. While storing uses a variety of datatypes this is not a fundamental concern. Recognizing and grouping are both related and challenging. The problems encountered with recognition and grouping are being investigated in the wider field of freeform digital ink. While this is outside the scope of this review we refer the interested reader to [32].

In addition to the individual steps, there is also the question of what order should these steps be applied. We have identified two general orders: process each annotation individually; and process all annotations each time a change is made. Each approach offers benefits and trade-offs. Processing each annotation individually is faster but information from other annotations can help with processing an annotation. Individual processing also fixes the annotation type at the time it is added; whereas for all annotation processing the type may change as other gestures are added thus potentially confusing the user. Thus this is an open area of research.

4.4. Adapting operations

The current literature has mixed results around the adapting operations. Repositioning by itself has well-defined solutions. The errors around repositioning are not due to the repositioning itself but because of errors in the adding operations. Improving the adding operations, either individually or together, will improve repositioning without any additional work. In contrast, both refitting and orphaning do require additional work. Also, these two areas are under-represented in the research. Refitting has only been implemented in four implementations and the focus has been on a very narrow set of annotation types (single line and multiple line). Of these four implementations, only one has looked at the user expectations. Orphaning has also been in four implementations and only one type of orphaning has been implemented. In addition, none of these have looked at user expectations. In the aligned field of text annotation, studies have found that users expect annotations to be available after their anchor has been deleted [11, 66]. Based on prior research this could take two forms. The first is to store all the orphaned annotations so the user can review them later [3]. An alternate approach would be to show an icon at the "best guess" location on the associated document [11]. Selecting this icon would then display the annotation. For each approach the user should be provided options on what to do with the orphaned annotation (e.g. delete, reposition, modify).

4.5. Annotation categories

In her original taxonomy Marshall [46] classified annotations along two dimensions: within-text vs. marginal or blank space; and telegraphic vs. explicit. During her investigation Marshall looked at annotations added to textbooks: one assumes for student study. Both the annotations and their underlying context are static. In contrast, digital documents are dynamic so the content underneath annotations can change. Thus Marshall's taxonomy, while still valid, is more limited for dynamic documents. Some annotations on dynamic documents do not fit in Marshall's four quadrants. For example, connector annotations for in both within-text and marginal and commands may be added anywhere on the document. Given these challenges we use an alternate form of taxonomy based on how the annotation would adapt in a digital environment.

This taxonomy builds on the work by Golovchinsky and Denoue and Bargeron and Moscovich. In their work [24, 8] they grouped annotations into three categories: single line, multiple line and complex[4]. Previously both connectors and commands were categorized as complex annotations. This is partly because these categories were used as a basis for automatically adapting annotations. Single line and multiple annotations were refitted as the underlying content changed while complex annotations were not. To these three categories we add connectors and commands (see Table 5).

Connectors are often used to link another annotation to a location in the underlying document [64, 20, 82]. Accordingly, they potentially have two an-

[3]Prior work indicts the associated context must also be stored [77].

[4]Both publications included similar annotation types in each category. The only exception is enclosures. Bargeron and Moscovich treated these primarily as single line annotations while Golovchinsky and Denoue treated them as multiple line.

chor points: one fixed to a location in the document and another associated with an annotation. An alternate form of connector is one that joins two sections of the content. These connectors also have two anchor points but both associated with the document. There has been no research published investigating how to automatically adapt connectors. Connectors should be repositioned like any other annotation; we also assume it would follow similar rules for orphaning. However refitting is a more interesting scenario: theoretically it would possible to refit connectors so these two anchor points move independently. There has been no research published on how this work work or, more importantly, on what the user expectations are. We postulate that there are two forms of refitting a connector. For connectors associated with another annotation the connector and associated annotation should be repositioned so the document anchor remains valid. For connectors associated with two different locations in the document the connector should be stretched or shrunk so the two anchor points remain in the correct locations.

Commands are often used for instructing the implementation to do something (e.g. erase or move content [84, 30, 21], move to another location [5, 6, 4, 81], link documents [74], etc.) Unlike most other annotations the implementation is expected to understand these annotations. One major area of research around commands is how to recognise them (see below). Unlike most other forms of annotation commands are transient and have only a short term lifetime. Again, there has been no research on automatically adapting command annotations. We also posit these would follow the same rules for repositioning and orphaning as other annotations. However given that commands are temporary the value of refitting them is questionable.

Combining together our two taxonomies (annotation types and annotation support operations) we can see the biggest gaps are refitting connector, command and complex annotations and orphaning. There are very few implementations that look at the technical complexities and none that investigate user expectations.

4.6. Annotation lifetime

The lifetime of an annotation is an important concept that, while evident from this review, is not widely discussed. There is a continuum of lifetimes; with three major points on the continuum: instantaneous, short term, and long term. Functional commands are instantaneous. The command is executed and the annotation discarded. Short term annotations have a limited lifetime. Once the annotation indicating that something needed editing has fulfilled its purpose it is removed. Long term annotations become part of the document; for example providing commentary or explanatory notes.

There is an interplay between lifetime and fluidity. In order to differentiate functional commands and digital ink many systems required the user to change modes (which is cognitively disruptive). This is because the software has difficulty reliably differentiating gesture classes. There is ongoing research into gesture recognition that may provide a solution; however currently there is a need to provide the software with a way separate commands from ink. There are a number of solutions suggested including using buttons (pen-based or separate), separate display areas, special gestures, pressure, and pen and touch. Each of these has its own limitations and strengths: which is most suitable is context dependent. Buttons require specialized hardware, and added user dexterity but have the advantage of certainty. Separate display areas uses screen real estate and requires a move in focus for the user but can provide a zoomed area for writing. Both special gestures and pressure require training (the user, the system or both) and recognition errors are still possible but results in a more fluid interaction. Pen and touch requires special hardware, bimanual interaction and recognition but has the most potential for providing fluid interaction and builds on our inherent bimanual abilities: for example a person may draw with pencil in one hand and an eraser in the other. Li et al. [38] investigated the performance of some of these approaches and found a bimanual approach (pen in preferred hand and button-push with non-preferred hand) was the fastest. This approach also had one of the lowest error rates and was preferred by most participants [38].

4.7. Other functionality

The systematic review has identified a wide range of other functionality that is provided in various projects (see Table 10). The most important of these are navigation and collaboration support. However the work in regards to both of these is immature and intersects with the related fields of computer supported collaborative work and document libraries respectively. These two areas, and the others identified in Table 10, warrant a specific literature review as more investigation is undertaken. This review could also go further and incorporate other functionality in other areas of annotation (i.e. text-based annotations).

4.8. User experience

Finally, of note is that the research into the technical issues with annotation are more advanced than the user experience. While the work of Marshall [46] laid an excellent foundation for how people annotate books, many of the studies reported here focus entirely on the technical issue. Those user studies that are reported, for example [55, 44, 77], are usually usability studies that evaluate the usability of the specific application without regard to the fundamental and theoretical principles. We could only find two studies [24, 8] that investigated user expectations on adapting annotations. While there has been work in this area for text-based annotation there is an urgent need for more work in this regard around freeform ink annotation.

In addition, most studies have focused on evaluating the effectiveness of their own implementation. Very few implementations attempt to generalise beyond their initial implementation[5]. While there are many solutions to technical challenges and interesting ideas for functionality, it is hard to generalise beyond the initial implementations. What works in one particular implementation, with its specific environment and objectives, may not work when transposed to another implementation. This may be a limitation of our field, where we focus on smaller units of work rather than exploring the bigger picture, that limits the transferability of our findings outside our field. This raises the question: are we leaving the bigger picture of how our work could benefit mankind to industry? This is a serious issue as industry has different objectives and driving motives which skews the long-term benefit of our research.

5. Future Directions

This review has identified several areas of investigation in future. These include:

(i) Investigating how both connectors and commands could be automatically adapted. This includes both the technical aspects and the user expectations, as well as investigating all three adapting operations.

(ii) Investigating the two overall approaches for adding annotations and the strengths and limitations of each approach. We posit that each

[5] XLibris is the main exception.

overall approach will be useful for different scenarios but we do not which approach is better or how this would be evaluated.

(iii) Improving the accuracy of each step for adding annotations. This includes reviewing these operations in the wider domain of freeform digital ink and their applicability to annotations.

(iv) Investigating how orphaned annotations can be handled. Again there is prior work in other domains that can be used as a starting point.

(v) Studying user expectations around freeform ink annotations in documents; especially for dynamic documents.

(vi) Reviewing the additional functionality provided by digital annotations.

6. Concluding Remarks

The motivation for this review was to determine what has been investigated for freeform digital ink annotations on text documents. Two research questions were formulated to guide the review. First, the operations needed to robustly add annotations; and second what support there is for automatically adapting annotations when a document changes. Using a systematic mapping study we present a taxonomy of current work.

Adding annotations to documents is well covered in the research. There are four operations used for adding annotations: grouping; recognising; anchoring; and storing. However there is not a common order to how these operations are used; instead there are variations based on the overall approach used and the level of user interaction provided. We also identified ten commonly recognised types of annotation. These are grouped into five categories based on their requirements for adding and adapting.

Automatic adapting of annotations has not been investigated widely. Repositioning is the most common adapting implementation, followed by orphaning and then refitting annotations. If the annotation is added robustly then repositioning occurs without additional work. The implementations that implement orphaning all work by deleting the annotation. The two implementations refit annotations only look at a reduced set of annotations: single-line and simple multi-line annotations. This is an area that needs additional investigation.

The most common type of human study is a usability study of how well an implementation performs but these do not improve the overall understanding of the underlying user expectations. There are few studies that investigate user expectations and only one that studied adapting annotations. This is a major gap in the current literature. We do not know how people will react to different types of changes or even how they might want an annotation to change. Nor are there technical solutions to many of the annotation styles commonly used. Future work addressing this gap digital ink annotation research should use the taxonomy presented here to describe how the research relates to the field.

Finally, one challenge in this study was identifying similar functionality due to the different terms used for the same thing. We recommend that a standardised set of terminology be used in future. In this study we suggest an initial set of definitions.

7. References

[1] A Adler, A Gujar, BL Harrison, K O'Hara, and A Sellen. A diary study of work-related reading: Design implications for digital reading devices. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '98, pages 241–248, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. doi: 10.1145/274644.274679.

[2] MJ Adler and C Van Doren. *How to Read a Book*. Simon and Schuster, New York, 1972.

[3] M Agrawala and M Shilman. DIZI: A digital ink zooming interface for document annotation. In MF Costabile and F Paternò, editors, *Human-Computer Interaction - INTERACT 2005*, volume 3585 of *Lect. Notes in Comput. Sci.*, chapter 9, pages 69–79. Springer Berlin Heidelberg, 2005. doi: 10.1007/11555261_9.

[4] R Anderson, L. McDowell, and B Simon. Use of classroom presenter in engineering courses. In *Proc. of the 35th Annu. Conf. on Front. in Educ.*, FIE '05, pages T2G–13, 2005. doi: 10.1109/FIE.2005.1611913.

[5] RJ Anderson, R Anderson, B Simon, SA Wolfman, T VanDeGrift, and K Yasuhara. Experiences with a tablet PC based lecture presentation system in computer science courses. *SIGCSE Bull.*, 36(1):56–60, 2004. doi: 10.1145/1028174.971323.

[6] RJ Anderson, C Hoyer, C. Prince, J. Su, F. Videon, and SA Wolfman. Speech, ink, and slides: The interaction of content channels. In *Proc. of the 12th Annu. ACM Int. Conf. on Multimedia*, MULTIMEDIA '04,

pages 796–803, New York, NY, USA, 2004. ACM. doi: 10.1145/1027527.1027713.

[7] E Ball, H Franks, J Jenkins, M McGrath, and J Leigh. Annotation is a valuable tool to enhance learning and assessment in student essays. *Nurse Educ. Today*, 29(3):284–291, 2009. doi: 10.1016/j.nedt.2008.10.005.

[8] D Bargeron and T Moscovich. Reflowing digital ink annotations. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '03, pages 385–393, New York, NY, USA, 2003. ACM. doi: 10.1145/642611.642678.

[9] K Bhardwaj, S Chaudhury, and SD Roy. Augmented paper system: A framework for user's personalized workspace. In *Proc. of the 4th Natl. Conf. on Comput. Vis., Pattern Recognit., Image Process. and Graph.*, pages 1–4, 2013. doi: 10.1109/NCVPRIPG.2013.6776182.

[10] P Brandl, M Haller, J Oberngruber, and C Schafleitner. Bridging the gap between real printouts and digital whiteboard. In *Proc. of the Work. Conf. on Adv. Vis. Interfaces*, AVI '08, pages 31–38, New York, NY, USA, 2008. ACM. doi: 10.1145/1385569.1385577.

[11] AJ Brush, A Bargeron, A Gupta, and JJ Cadiz. Robust annotation positioning in digital documents. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '01, pages 285–292, New York, NY, USA, 2001. ACM. doi: 10.1145/365024.365117.

[12] D Cabral and N Correia. Pen-based video annotations: A proposal and a prototype for tablet PCs. In T Gross, T Gulliksen, P Kotzé, L Oestreicher, P Palanque, RO Prates, and M Winckler, editors, *Human-Computer Interaction - INTERACT 2009*, volume 5727 of *Lect. Notes in Comput. Sci.*, pages 17–20. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-03658-3_5.

[13] JJ Cadiz, A Gupta, and J Grudin. Using web annotations for asynchronous collaboration around documents. In *Proc. of the 2000 ACM Conf. on Comput. Support. Cooperative Work*, CSCW '00, pages 309–318, 359002, 2000. ACM. doi: 10.1145/358916.359002.

[14] RG Cattelan, C Teixeira, H Ribas, E Munson, and MGC Pimentel. Inkteractors: Interacting with digital ink. In *Proc. of the 2008 ACM Symp. on Appl. Comput.*, SAC '08, pages 1246–1251, New York, NY, USA, 2008. ACM. doi: 10.1145/1363686.1363973.

[15] S Chandrasekar, JG Tront, and JC Prey. WriteOn1.0: A tablet PC-based tool for effective classroom instruction. *SIGCSE Bull.*, 41(3): 323–327, 2009. doi: 10.1145/1595496.1562975.

[16] SH Chang, X Chen, RA Priest, and B Plimmer. Issues of extending the user interface of integrated development environments. In *Proc. of the 9th ACM SIGCHI New Zealand Chapter's Int. Conf. on Hum.-Comput. Interact.*, CHINZ '08, pages 23–30, New York, NY, USA, 2008. ACM. doi: 10.1145/1496976.1496980.

[17] MA Chatti, T Sodhi, M Specht, R Klamma, and R Klemke. u-Annotate: An application for user-driven freeform digital ink annotation of E-Learning content. In *Proc. of the 6th Int. Conf. on Adv. Learn. Technol.*, pages 1039–1043, 2006. doi: 10.1109/ICALT.2006.1652624.

[18] N Chen, F Guimbretière, and A Sellen. Designing a multi-slate reading environment to support active reading activities. *ACM Trans. Comput.-Hum. Interact.*, 19(3):18:1–18:35, 2012. doi: 10.1145/2362364.2362366.

[19] N Chen, F Guimbretière, and A Sellen. Graduate student use of a multi-slate reading system. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '13, pages 1799–1808, New York, NY, USA, 2013. ACM. doi: 10.1145/2470654.2466237.

[20] X Chen and B Plimmer. CodeAnnotator: Digital ink annotation within Eclipse. In *Proc. of the 19th Australas. Conf. on Comput.-Hum. Interact.*, OZCHI '07, pages 211–214, New York, NY, USA, 2007. ACM. doi: 10.1145/1324892.1324935.

[21] K Conroy, D Levin, and F Guimbretière. Proofrite: A paper-augmented word processor. In *Demo Session of UIST*. ACM, 2004.

[22] M Dontcheva, SM Drucker, and MF Cohen. V4V: A view for the viewer. In *Proceedings of the 2005 Conference on Designing for User eXperience*, DUX '05, New York, NY, USA, 2005. AIGA: American Institute of Graphic Arts.

[23] RL Fowler and AS Barker. Effectiveness of highlighting for retention of text material. *J. of Appl. Psychol.*, 59(3):358, 1974.

[24] G Golovchinsky and L Denoue. Moving markup: Repositioning freeform annotations. In *Proc. of the 15th Ann. ACM Symp. on User Interface Softw. and Technol.*, UIST '02, pages 21–30, New York, NY, USA, 2002. ACM. doi: 10.1145/571985.571989.

[25] G Golovchinsky and CC Marshall. Hypertext interaction revisited. In *Proc. of the 11th ACM Conf. on Hypertext and Hypermedia*, HYPERTEXT '00, pages 171–179, New York, NY, USA, 2000. ACM. doi: 10.1145/336296.336358.

[26] G Golovchinsky and CC Marshall. Hypertext interactivity: from choice to participation. *New Rev. of Hypermedia and Multimedia*, 6(1):169–196, 2000. doi: 10.1080/13614560008914722.

[27] G Golovchinsky, MN Price, and BN Schilit. From reading to retrieval: Freeform ink annotations as queries. In *Proc. of the 22Nd Ann. Int. ACM SIGIR Conf. on Res. and Dev. in Inf. Retr.*, SIGIR '99, pages 19–25, New York, NY, USA, 1999. ACM. doi: 10.1145/312624.312637.

[28] M Götze, S Schlechtweg, and T Strothotte. The intelligent pen: Toward a uniform treatment of electronic documents. In *Proc. of the 2nd Int. Symp. on Smart Graph.*, SMARTGRAPH '02, pages 129–135, New York, NY, USA, 2002. ACM. doi: 10.1145/569005.569024.

[29] F Guimbretière. Paper augmented digital documents. In *Proc. of the 16th Ann. ACM Symp. on User Interface Softw. and Technol.*, UIST '03, pages 51–60, New York, NY, USA, 2003. ACM. doi: 10.1145/964696.964702.

[30] G Hardock, G Kurtenbach, and W Buxton. A marking based interface for collaborative writing. In *Proc. of the 6th Ann. ACM Symp. on User Interface Softw. and Technol.*, UIST '93, pages 259–266, New York, NY, USA, 1993. ACM. doi: 10.1145/168642.168669.

[31] K Hinckley, X Bi, M Pahud, and B Buxton. Informal information gathering techniques for active reading. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '12, pages 1893–1896, New York, NY, USA, 2012. ACM. doi: 10.1145/2207676.2208327.

[32] G Johnson, MD Gross, J Hong, and EY Do. *Computational Support for Sketching in Design: A Review*, volume 2. Now Publishers Inc., Hanover, MA, USA, 2009. doi: 10.1561/1100000013.

[33] M Kam, J Wang, A Iles, E Tse, J Chiu, D Glaser, O Tarshish, and J Canny. LiveNotes: A system for cooperative and augmented note-taking in lectures. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '05, pages 531–540, New York, NY, USA, 2005. ACM. doi: 10.1145/1054972.1055046.

[34] R Kawase, E Herder, and W Nejdl. A comparison of paper-based and online annotations in the workplace. In U Cress, V Dimitrova, and M Specht, editors, *Learn. in the Synerg. of Mult. Discipl.*, volume 5794 of *Lect. Notes in Comput. Sci.*, pages 240–253. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-04636-0_23.

[35] B Kitchenham and S Charters. Guidelines for performing systematic literature reviews. Technical report, Keele University and Durham University Joint Report, 2007.

[36] B Kitchenham, R Pretorius, D Budgen, O Brereton, M Turner, M Niazi, and S Linkman. Systematic literature reviews in software engineering a tertiary study. *Inf. and Softw. Technol.*, 52(8):792–805, 2010. doi: 10.1016/j.infsof.2010.03.006.

[37] SR Levine and SF Ehrlich. The Freestyle system. In A Klinger, editor, *Hum.-Mach. Interactive Syst.*, Lang. and Inf. Syst., pages 3–21. Springer US, 1991. doi: 10.1007/978-1-4684-5883-1_1.

[38] Y Li, K Hinckley, Z Guan, and JA Landay. Experimental analysis of mode switching techniques in pen-based user interfaces. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '05, pages 461–470, New York, NY, USA, 2005. ACM. doi: 10.1145/1054972.1055036.

[39] C Liao, F Guimbretière, R Anderson, N Linnell, C Prince, and V Razmov. PaperCP: Exploring the integration of physical and digital affordances for active learning. In C Baranauskas, P Palanque, J Abascal, and SDJ Barbosa, editors, *Human-Computer Interaction INTERACT 2007*, volume 4663 of *Lect. Notes in Comput. Sci.*, pages 15–28. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-74800-7_2.

[40] G Liao, F Guimbretière, and K Hinckley. PapierCraft: A command system for interactive paper. In *Proc. of the 18th Ann. ACM Symp. on User Interface Softw. and Technol.*, UIST '05, pages 241–244, New York, NY, USA, 2005. ACM. doi: 10.1145/1095034.1095074.

[41] G Liao, F Guimbretière, K Hinckley, and J Hollan. PapierCraft: A gesture-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.*, 14(4):18:1–18:27, 2008. doi: 10.1145/1314683.1314686.

[42] L Lichtschlag and J Borchers. CodeGraffiti: Communication by sketching for pair programmers. In *Adjunct Proc. of the 23nd Annu. ACM Symp. on User Interface Softw. and Technol.*, UIST '10, pages 439–440, New York, NY, USA, 2010. ACM. doi: 10.1145/1866218.1866260.

[43] Z Liu. Reading behavior in the digital environment: Changes in reading behavior over the past ten years. *J. of Doc.*, 61(6):700–712, 2005.

[44] WE Mackay, G Pothier, C Letondal, K Bøegh, and H. E. Sørensen. The missing link: Augmenting biology laboratory notebooks. In *Proc. of the 15th Annu. ACM Symp. on User Interface Softw. and Technol.*, UIST '02, pages 41–50, New York, NY, USA, 2002. ACM. doi: 10.1145/571985.571992.

[45] S Marinai. Reflowing and annotating scientific papers on eBook readers. In *Proc. of the 2013 ACM Symp. on Doc. Eng.*, DocEng '13, pages 241–244, New York, NY, USA, 2013. ACM. doi: 10.1145/2494266.2494311.

[46] CC Marshall. Annotation: From paper books to the digital library. In *Proc. of the 2nd ACM Int. Conf. on Digit. Libr.*, DL '97, pages 131–140, New York, NY, USA, 1997. ACM. doi: 10.1145/263690.263806.

[47] CC Marshall and AJB Brush. From personal to shared annotations. In *CHI '02 Ext. Abstr. on Hum. Factors in Comput. Syst.*, CHI EA '02, pages 812–813, New York, NY, USA, 2002. ACM. doi: 10.1145/506443.506610.

[48] CC Marshall, MN Price, G Golovchinsky, and BN Schilit. Collaborating over portable reading appliances. *Pers. Technol.*, 3(1-2):43–53, 1999. doi: 10.1007/BF01305319.

[49] CC Marshall, MN Price, G Golovchinsky, and BN Schilit. Designing e-Books for legal research. In *Proc. of the 1st ACM/IEEE-CS Joint Conf. on Digit. Libr.*, JCDL '01, pages 41–48, New York, NY, USA, 2001. ACM. doi: 10.1145/379437.379445.

[50] F Matulic and MC Norrie. Supporting active reading on pen and touch-operated tabletops. In *Proc. of the Int. Work. Conf. on Adv. Visual Interfaces*, AVI '12, pages 612–619, New York, NY, USA, 2012. ACM. doi: 10.1145/2254556.2254669.

[51] A Mazzei, J Blom, L Gomez, and P Dillenbourg. Shared annotations: The social side of exam preparation. In D Hernández-Leo, T Ley, R Klamma, and A Harrer, editors, *Scaling up Learning for Sustained Impact*, volume 8095 of *Lect. Notes in Comput. Sci.*, pages 205–218. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-40814-4_17.

[52] MR Morris, AJB Brush, and BR Meyers. Reading revisited: Evaluating the usability of digital display surfaces for active reading tasks. In *2nd Annu. IEEE Int. Workshop on Horiz. Interactive Hum.-Comput. Syst.*, pages 79–86, 2007. doi: 10.1109/TABLETOP.2007.12.

[53] K O'Hara and A Sellen. A comparison of reading paper and on-line documents. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '97, pages 335–342, New York, NY, USA, 1997. ACM. doi: 10.1145/258549.258787.

[54] C Okoli and K Schabram. A guide to conducting a systematic literature review of information systems research. *Sprouts: Work. Papers on Inf. Syst.*, 10(26), 2010.

[55] DR Olsen, T Taufer, and JA Fails. ScreenCrayons: Annotating anything. In *Proc. of the 17th Annu. ACM Symp. on User Interface Softw. and Techn.*, UIST '04, pages 165–174, New York, NY, USA, 2004. ACM. doi: 10.1145/1029632.1029663.

[56] J Pearson and G Buchanan. Real-time document collaboration using iPads. In *Proc. of the 3rd Workshop on Res. Adv. in Large Digit. Book Repos. and Complement. Media*, BooksOnline '10, pages 9–14, New York, NY, USA, 2010. ACM. doi: 10.1145/1871854.1871859.

[57] B Plimmer. A comparative evaluation of annotation software for grading programming assignments. In *Proc. of the 11th Australasian Conf. on User Interface*, AUIC '10, pages 14–22, Darlinghurst, Australia, Australia, 2010. Australian Computer Society, Inc.

[58] B Plimmer and M Apperley. Making paperless work. In *Proc. of the 8th ACM SIGCHI New Zealand Chapter's Int. Conf. on Comput.-Hum, Interaction*, CHINZ '07, pages 1–8, New York, NY, USA, 2007. ACM. doi: 10.1145/1278960.1278961.

[59] B Plimmer and P Mason. A pen-based paperless environment for annotating and marking student assignments. In *Proc. of the 7th Australasian User Interface Conf.*, AUIC '06, pages 37–44, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[60] B Plimmer, J Grundy, J Hosking, and R Priest. Inking in the IDE: Experiences with pen-based design and annotation. In *IEEE Symp. on Vis. Lang. and Hum.-Centric Comput.*, pages 111–115, 2006. doi: 10.1109/VLHCC.2006.28.

[61] B Plimmer, SH Chang, M Doshi, L Laycock, and N Seneviratne. iAnnotate: Exploring multi-user ink annotation in web browsers. In *Proc. of the 11th Australasian Conf. on User Interface*, AUIC '10, pages 52–60, Darlinghurst, Australia, Australia, 2010. Australian Computer Society, Inc.

[62] MN Price, G Golovchinsky, and BN Schilit. Linking by inking: Trailblazing in a paper-like hypertext. In *Proc. of the 9th ACM Conf. on Hypertext and Hypermedia*, HYPERTEXT '98, pages 30–39, New York, NY, USA, 1998. ACM. doi: 10.1145/276627.276631.

[63] MN Price, BN Schilit, and G Golovchinsky. XLibris: The active reading machine. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '98, pages 22–23, New York, NY, USA, 1998. ACM. doi: 10.1145/286498.286510.

[64] R Priest and B Plimmer. RCA: Experiences with an IDE annotation tool. In *Proc. of the 7th ACM SIGCHI New Zealand Chap. Int. Conf. on Comput.-Hum. Interaction*, CHINZ '06, pages 53–60, New York, NY, USA, 2006. ACM. doi: 10.1145/1152760.1152767.

[65] S Ramachandran and R Kashi. An architecture for ink annotations on web documents. In *Proc. of the 7th Int. Conf. on Doc. Analysis and Recognition*, pages 256–260 vol.1, 2003. doi: 10.1109/ICDAR.2003.1227669.

[66] R Sanderson and H Van de Sompel. Making web annotations persistent over time. In *Proc. of the 10th Ann. Joint Conf. on Digit. Lib.*, JCDL '10, pages 1–10, New York, NY, USA, 2010. ACM. doi: 10.1145/1816123.1816125.

[67] BN Schilit, G Golovchinsky, and MN Price. Beyond paper: Supporting active reading with free form digital ink annotations. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '98, pages 249–256, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. doi: 10.1145/274644.274680.

[68] BN Schilit, MN Price, and G Golovchinsky. Digital library information appliances. In *Proc. of the 3rd ACM Conf. on Digit. Lib.*, DL '98, pages 217–226, New York, NY, USA, 1998. ACM. doi: 10.1145/276675.276700.

[69] AJ Sellen and R Harper. *The myth of the paperless office*. MIT Press, Cambridge, Mass, 2002.

[70] M Shilman and Z Wei. Recognizing freeform digital ink annotations. In S Marinai and A Dengel, editors, *Doc. Anal. Syst.*, volume 3163 of *Lect. Notes in Comput. Sci.*, pages 322–331. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-540-28640-0_30.

[71] F Shipman, MN Price, CC Marshall, and G Golovchinsky. Identifying useful passages in documents based on annotation patterns. In T Koch and I Sølvberg, editors, *Res. and Adv. Technol. for Digit. Lib.*, volume 2769 of *Lect. Notes in Comput. Sci.*, pages 101–112. Springer Berlin Heidelberg, 2003. doi: 10.1007/978-3-540-45175-4_11.

[72] B Signer and MC Norrie. PaperPoint: A paper-based presentation and interactive paper prototyping tool. In *Proc. of the 1st Int. Conf. on Tangible and Embedded interaction*, TEI '07, pages 57–64, New York, NY, USA, 2007. ACM. doi: 10.1145/1226969.1226981.

[73] ML Simpson and SL Nist. Textbook annotation: An effective and efficient study strategy for college students. *J. of Read.*, 34(2):122–129, 1990. doi: 10.2307/40032053.

[74] J Steimle. CoScribe: Integrating paper and digital documents for collaborative knowledge work. volume 2, pages 174–188, 2009. doi: 10.1109/TLT.2009.27.

[75] J Steimle. *Collaborative Cross-media Annotation of Documents*, pages 103–126. Human-Computer Interaction Series. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-20276-6_5.

[76] J Steimle, O Brdiczka, and M Muhlhauser. CoScribe: Integrating paper and digital documents for collaborative knowledge work. *Learning Technologies, IEEE Transactions on*, 2(3):174–188, 2009.

[77] CJ Sutherland and B Plimmer. vsInk: Integrating digital ink with program code in visual studio. In *Proceedings of the Fourteenth Australasian User Interface Conference - Volume 139*, AUIC '13, pages 13–22, Darlinghurst, Australia, Australia, 2013. Australian Computer Society, Inc.

[78] CJ Sutherland, A Luxton-Reilly, and B Plimmer. An observational study of how experienced programmers annotate program code. In J Abascal, S Barbosa, M Fetter, T Gross, P Palanque, and M Winckler, editors, *Human-Computer Interaction INTERACT 2007*, volume 9297 of *Lect. Notes in Comput. Sci.*, pages 177–194. Springer International Publishing, 2015. doi: 10.1007/978-3-319-22668-2_15.

[79] JG Tront, V Eligeti, and J Prey. Classroom presentations using tablet PCs and WriteOn. In *Annu. Frontiers in Ed. Conf*, pages 1–5, 2006. doi: 10.1109/FIE.2006.322336.

[80] JG Tront, V Eligeti, and J Prey. WriteOn: A tool to support teaching software engineering. In *19th Conf. on Softw. Eng. Ed. and Training Workshops*, pages 8–8, 2006. doi: 10.1109/CSEETW.2006.25.

[81] KN Truong, GD Abowd, and JA Brotherton. Personalizing the capture of public experiences. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, pages 121–130, New York, NY, USA, 1999. ACM. doi: 10.1145/320719.322593.

[82] X Wang and S Raghupathy. Ink annotations and their anchoring in heterogeneous digital documents. In *9th Int. Conf. on Doc. Anal. and Recognition*, volume 1, pages 163–167, 2007. doi: 10.1109/ICDAR.2007.4378696.

[83] X Wang, M Shilman, and S Raghupathy. Parsing ink annotations on heterogeneous documents. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 43–50. Eurographics Association, 2006.

[84] N Weibel, A Ispas, B Signer, and MC Norrie. PaperProof: A paper-digital proof-editing system. In *CHI '08 Ext. Abstr. on Hum. Factors in Comput. Syst.*, CHI EA '08, pages 2349–2354, New York, NY, USA, 2008. ACM. doi: 10.1145/1358628.1358682.

[85] H Wu, SJH. Yang, and Y Su. Free-form annotation tool for collaboration. In *IEEE Int. Conf. on Sensor Networks, Ubiquitous and Trustworthy Comput.*, SUTC '08, pages 555–560, 2008. doi: 10.1109/SUTC.2008.60.

[86] D Yoon, N Chen, and F Guimbretière. TextTearing: Opening white space for digital ink annotation. In *Proc. of the 26th Annu. ACM Symp. on User Interface Softw. and Technol.*, UIST '13, pages 107–112, New York, NY, USA, 2013. ACM. doi: 10.1145/2501988.2502036.

[87] R Zeleznik, T Miller, C Li, and JJ Laviola Jr. MathPaper: Mathematical sketching with fluid support for interactive computation. In A Butz, B Fisher, A Krüger, P Olivier, and M Christie, editors, *Smart Graphics*, volume 5166 of *Lect. Notes in Comput. Sci.*, pages 20–32. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-85412-8_3.

[88] S Zyto, DR Karger, MS Ackerman, and S Mahajan. Successful classroom deployment of a social document annotation system. In *Proc. of the SIGCHI Conf. on Hum. Factors in Comput. Sys.*, CHI '12, pages 1883–1892, New York, NY, USA, 2012. ACM. doi: 10.1145/2207676.2208326.