A Comparative Evaluation of Annotation Software for Grading Programming Assignments

Beryl Plimmer
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland 1142, New Zealand
beryl@cs.auckland.ac.nz

Abstract

Commenting on a student's computer program with red pen ink annotations is not possible with current software and paper program 'listings' are a relic of a bygone era. Yet ink annotations are the easiest way to provide rich feedback to the student. We have developed and evaluated Penmarked as a software solution to this problem. It supports free-form ink annotations and, importantly, associated marking tasks of gathering and returning assignments and recording grades. The evaluation against paper and digital marking systems showed it to be faster and more effective. From a wider perspective Penmarked demonstrates the intricacies of providing totally paperless environment.

Keywords: Digital Ink Annotation, task support, assignment grading.

1 Introduction

Annotating student assignments with a red pen is a basic recording mechanism for teachers. The annotations provide commentary to the student on the marker's response to the work and also provide 'backtalk' (Goldschmidt 1999) to the marker to assist with grading decisions. Annotating paper scripts is simple, however we are increasingly moving to digital environments where there is no paper copy of the work. In these environments students submit their work into a digital drop-box, the marker reads and marks it from this repository and returns feedback and grades to the student electronically. Digital alternatives to paper annotation include a marking schedule which incorporates space for off-document comments or digital annotation tools to provide in-situ comments. Either of these alternatives presents problems to the marker and student. Off-document comments are slow to construct and understand, and usually restricted to text. Digital annotation is available in some software applications (such as adobe acrobat) but interfacing these general tools effectively to student management systems and incorporating grade recording is difficult. Furthermore computer program code presents extra challenges because of its non-linear structure and multi-file nature.

There is a conflict here, ink annotations are the traditional and, we believe, easiest way to provide rich feedback to students. However electronic submission systems make ink annotation of paper copies difficult to support. There are many, most locally developed, systems to support off document comments as a part of assignment marking (our department has at least 3). But writing off document comments is tedious and therefore often neglected. As a consequence of these difficulties students are missing out on personalized, meaningful feedback on a critical part of their learning experience.

Computer programs pose some unique requirements for annotation and marking support. First, the non-lineal nature of programs means it makes no sense to read from the beginning to the end of a program (as one does an essay). Second, as a part of the evaluation process most markers (teachers or teaching assistants) compile and execute the program: to do this they must work with a digital copy of the assignment. Hence electronic submission of programming assignments has become the norm. Digital drop-boxes have the added advantages of supporting diverse teaching modalities; in-class, distance education, e-learning etc.

Usable and affordable pen input devices, in particular tablet PCs, mean that we now have the hardware to support input of ink annotations directly onto the document surface. Here we present Penmarked, a software solution to this problem, and its comprehensive evaluation from the teachers, markers and students perspectives.

2 Motivation

Consider a TA 50 marking programming assignments. They need to check each program against the assignment requirements, provide feedback to the student, record the grade in the administration system and send the comments and grade to the student. The most difficult part of this task with current systems is providing feedback to the student, so it is often ignored or poorly done. Yet personalized feedback is highly valued by students.

The design focus of Penmarked is to provide and efficient and effective environment for annotating and marking of computer programming assignments. In this paper we start with the educational case for good student feedback mechanisms before reviewing existing work on ink annotation and assignment annotation tools. This is followed by a description of the design and implementation of Penmarked. Then the evaluation methodology and results are described. In the discussion

section we reflect on the efficacy of our approach and lessons learnt including the requirements for program marking compared with the more general needs of paperless environments. The conclusions summarize our project and suggest directions for future work.

For clarity we include a short glossary here: *assessment*; we use to mean a class set of student assignments; *assignment*, is an individual student's work; *marking schedule* is a table of marking criteria and the mark allocated to each criterion, others may refer to this as a marking scheme or rubric (see Figure 2 for example).

3 Related Work

Many organizations are currently attempting to reduce or eliminate paper (Sellen and Harper 2002). Computer programs are, perhaps, the best example of a document type that has evolved to be best suited to digital environments. The folders of paper copies of program listings that existed in all programming shops have been replaced by on-line libraries as paper copies of large object oriented programs with their multiple classes and files make no sense. Programming integrated development environments (IDEs) and supporting tools are designed specifically to support program document management and other programming tasks. However they do not support assignment marking.

Educationalists agree that active learning, where the student is required to do something, is more effective that passive learning, where the student simply observes. In programming classes students are routinely set formative and summative programming tasks to actively engage them in learning correct programming techniques. To complete the feedback loop the work is reviewed and, if it is a summative assignment, a grade allocated. The reviewer's comments and grade assist the students to reflect on what they have done right and wrong and thus gain a better understanding of the subject. Thus the teacher and students together complete the learning cycle (Figure 1)

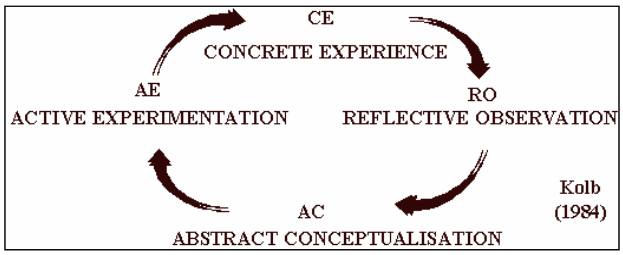

Figure 1 Kolb's Learning Cycle (from Greenaway 2003)

The marker's red ink annotations have traditionally decorated students' paper assignments. Ink annotations are rich and expressive due to their free-format, (Marshall 1997; Marshall, Price et al. 2001). Ink annotations also have a part to play in supporting active reading (Schilit, Golovchinsky et al. 1998). One of the purposes of making annotations is to help the reader engage more deeply in the material. These annotations also direct subsequent readers to points of interest (Shipman, Price et al. 2003). There is a well understood, yet informal, code for shared annotations such as ticks, crosses and enclosing loops with or without comments or marks attached and ideograms such as ☺. Studies comparing comprehension of annotated and un-annotated documents suggest that annotation is an effective aid to learning (Shipman, Price et al. 2003). From these studies we can conclude that annotation is useful for the marker to support his/her active reading of the document, and to the student to direct him/her to parts of the assignment that the marker thought were noteworthy.

One of the first tools developed to explore digital ink annotation was Wang Freestyle (Francik 1995). It provides the user with simple free-form ink annotation over a static page. XLibris (Schilit, Golovchinsky et al. 1998) was developed to offer users an active reading experience, with a main goal of addressing the tangibility challenges of reading online documents. It provides users with an interface and features similar to that of paper. While some word processors support ink comments and change tracking these environments are clumsy for multi-file programs. None of these general tools are supportive of our marker with 50 programming assignments to review and grade.

There are many automatic program marking programs, these are good for marking algorithm type problems where there is one correct answer. However many programming assignments include elements of user interface design and are designed to challenge the student to choose between alternative 'good solutions', making them unsuitable for automated marking.

We have identified three related annotation or marking tools. Marktool (Heinrich, Wang et al. 2003; Heinrich and Lawn 2004) supports annotation of assignments by use of drag-and-drop shapes and text boxes. Gild (Myers, Hargreaves et al. 2004) provides marking functionality within the Eclipse IDE but does not support digital ink or text annotation. Alongside this project we have explored annotation inside IDEs (Priest and Plimmer 2006; Chen and Plimmer 2007; Chang, Chen et al. 2008), we were unable to build a functioning clear annotation layer in either visual studio or Eclipse. Our current solution is to copy the document to be annotated and placed it as a background to the annotation pane; however this is not a very robust solution.

Digital ink annotation is technically challenging. The basic requirement is to collect ink input, then display and hold it in the correct place on the related document while the document is repositioned or resized. If the underlying document is dynamic then support for reflowing and reshaping of digital ink as the underlying layout changes is required. Margin bars, circles and underlines, must stretch or shrink with layout changes through font resizing, zooming or varying device characteristics (Golovchinsky and Denoue 2002). Annotations should also reshape when underlying text splits over line breaks and page breaks (Ramachandran and Kashi 2003). In this application the students' documents are assumed to be finished therefore ink reflow is not required. However, even without reflow support, digital ink annotation continues to be technically challenging because the ink must exist in a separate layer to the original document

and standard interface components do not support this notion.

4 Penmarked Software

The goal of Penmarked is to fully support the marking and annotating of students' assignments. It is written in C# for the Microsoft Tablet OS and relies on the inbuilt recognition engine for writing recognition. Here we describe its pertinent features, more technical descriptions are available elsewhere (Plimmer and Mason 2004; Mason and Plimmer 2005; Plimmer and Mason 2006). The software described here is the third prototype of Penmarked. The first (Plimmer and Mason 2004) solved the technical ink annotation challenges and supported basic collection and return of work. As a result of our usability testing (Mason and Plimmer 2005) a number of small, but important, interaction changes were made to the system. Feedback from users has resulted in additional functionality, such as the comment pane. The version reported in (Plimmer and Mason 2006) was used for the evaluation study reported here.

We use a scenario of a teacher with a class set of 50 programming assignments to mark to describe Penmarked. Before starting the teacher creates a new master marking schedule for the assessment. A wizard (Figure 2) is available from the edit menu to support its construction or alternatively the xml file can be edited directly. Another setup step is to decide which files are to be displayed in the annotation panel. Full flexibility based on file names and extensions with wild cards to include or exclude is in the options menu. This is particularly useful for some programming environments that include a number of management files that are not of interest to the teacher

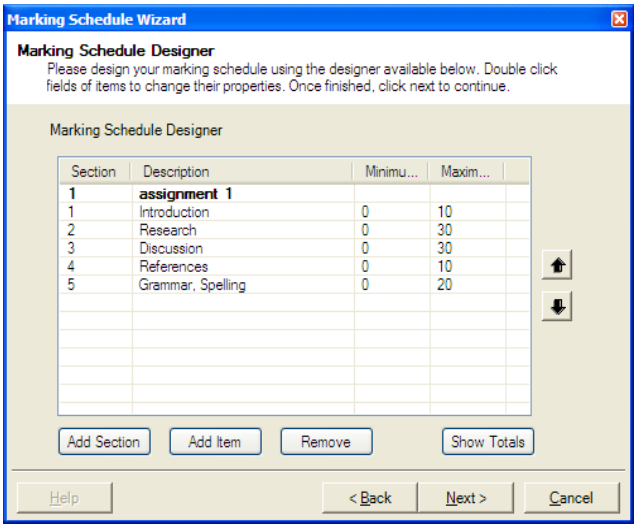

Figure 2 Marking Schedule Construction Wizard.

The final pre-marking step is to load the students' files. From the file menu the user can set up or edit an assessment (a class set of marking). The assessment data includes identification and date information, the marking schedule and folder locations of the student assignments.

The student assignments can then be opened for marking. Penmarked parses the location for the student files specified above looking for student data files. These files must contain the student identifiers and the names and types of the associated student's assignment files. Such files are common in drop boxes or student management systems but each has its own format. Currently we support two formats a simple text file and xml file; however Penmarked provides add-in functionality to extend the file types and structures for these files. All the student assignments found in the specified directory structures are listed in the left pane shown in Figure 3. The teacher can then commence marking by taping on a student identifier in the list pane on the left of Figure 3.

On the first opening of a student's assignment Penmarked checks for submitted zip files. If one is found a subdirectory named with the students identifier is created below the location of the zip file and the contents of the zip file are placed into this subdirectory. An individual copy of the marking schedule is also created for the newly opened student assignment and displayed in the bottom window pane as shown in Figure 3. On subsequent opening the unzipping is not required and the student's personal marking schedule is displayed including any marks already entered.

The annotation pane displays all the files that meet the filter specification, each in a separate tab. The marker can now peruse the assignment. In the annotation pane the marker can ink and erase freely over the assignment. The ink is placed on a transparent layer that lies above a rich textbox, thus the text is inaccessible to the marker. Making the text inaccessible was a design decision we made after talking with students and teachers, it protects the student's work from unintentional changes by the marker. The implementation challenges with this were many and varied, most difficult was scrolling both layers synchronously as it is difficult to access the appropriate scrolling methods on the lower layer (this is now simpler with Microsoft Presentation Foundation).

If the marker wants to run the program they can directly access the folder containing the student's files by clicking on the folder icon on the tool bar.

As marking progresses the teacher can enter marks against an item in the schedule by first selecting the item row with a tap or click and then either write the mark into the box in the right bottom corner of Figure 3 or enter the mark via a keyboard. If ink is entered in the writing box it is recognized as soon as another item in the schedule is selected or after a short time delay. The OS recognition engine using the number factoid restriction (which limits recognition results to digits and numeric symbols) is used for recognition. The data is validated against the minimum and maximum values. Valid data is saved into the schedule; the box flashes red if recognition fails to produce a valid result. In our various trials two users experienced problems with recognition errors: one who was more accustomed to writing Chinese formed his '5' more like an 's', a little training solved this problem; the other placed decimal points quite high between numerals '5.6' this is interpreted as a subtract symbol.

If the teacher wants to add some general comments a small comments pane can be opened on the right-hand side of the main window. In addition to the three main

panes the icons initiate frequently used functions of open, save, export, ink and erase.

When marking of a student's assignment is completed the associated student item can be ticked in the list. A right click on a student item in the list pane displays a menu of less frequent tasks. There is access to functionality such as adding a file to the student's assignment or cleaning all the annotations from the assignment. There are also options to mark the item as 'in progress' or 'recheck', these options change the colour of the list item to blue and red respectively.

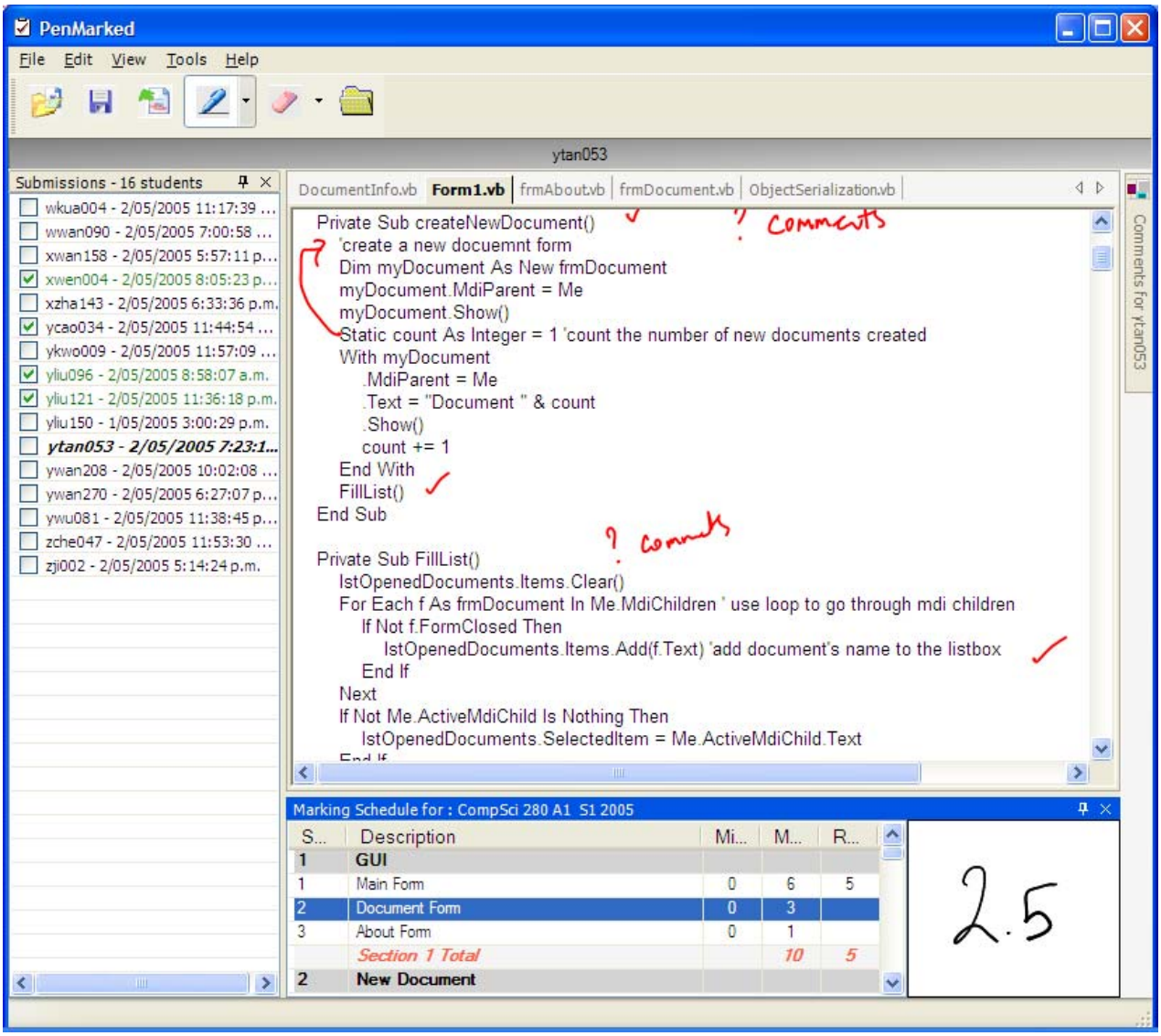

Figure 4 Main Penmarked interface

Once all the students' assignments have been marked there is a number of post-marking task required to return the assignments to the students and export the marks. There is a batch process to returned annotated pdf copies of the assignments to students. The marker simply ensures all assignments to be returned are selected in the list and selects 'return pdfs' from the file menu. Penmarked generates a pdf for each student that contains their marking schedule, general comments and annotated assignment and emails it to the student. A pdf can be generated independently of emailing. Similarly the detailed and total marks can be exported to an xml file, from which they can be imported into most standard software applications and student management systems.

Thus the marking task it completed. We contend that for efficiency marking software must support the entire marking process: gathering up the assignments; supporting examining, annotating and grading each assignment; returning the work to students and filing the grades. All in a time effective and easy-to-use manner.

The tasks supported by Penmarked for marking an assessment are:

1. Set up a marking schedule
2. Set up an assessment task
3. Gather up assignments
4. Examination, annotation and grading of assignments
5. Return work to students
6. Export of grades

During development Penmarked has been usability tested (Mason and Plimmer 2005) and trialled on a range of assessments including .Net programs, Java programs and essays. To evaluate the efficacy of Penmarked we

conducted a large evaluation study. The details of the study and results are described below.

5 Method

Our goal with the evaluation study was to consider the efficacy of Penmarked from the point-of-view of teachers, markers and students. We conducted a comparative study across three assessments of a second year programming class, applying three different marking treatments to each assessment in a Latin squares arrangement. Quantitative data was gathered from assignment marking while qualitative data was garnered from students, markers and teachers.

The three marking treatments are: traditional paper based marking, on-line marking where marks are recorded in a database system, and Penmarked. We will refer to these treatments as paper, database and Penmarked. The class consisted of approximately 200 students so approximately 600 individual pieces of work were marked. Six teaching assistants (TAs) marked one sixth of the assignments for each assessment (~33). Each student had one piece of work marked via each treatment and each TA marked using each treatment. The student/TA allocations were changed with each assessment so that a TA only marked one piece of work from each student. Therefore the study has marking method as the independent variable and, it is a between-subjects study and each TA and student marked, or had marked, a different assessment for each treatment.

The hypotheses to be tested were

- that there would be a difference in time required to mark using the different methods – with a corresponding null hypothesis that there would be no time difference.
- that the range of grades was consistent across all treatments;
- that there would be a difference in the number an of annotation or comments between the in treatments
- that there would be a difference in marker and student satisfaction between treatments.

The assessments varied in difficulty and length: the number of program classes (which equates to files) and lines of code in the model answers are, respectively 1/74, 3/198 and 5/435. As per our usual practice, for each assignment the markers were provided with a model answer, marking schedule and participated in two discussions about the marking (one before any marking and one after 3-4 assignments had been marked). Many students take considerably more lines of code to write a program than the model answers, sometimes twice as much. The paper copies of the assessments 1-3 were respectively approximately 2-3, 10-12, 15-25 pages. For the paper treatment the markers used the paper copy along with the digital copy of the assignment and the IDE for marking. For the database treatment the markers were interacting with the database forms, a digital copy of the assignment and the IDE. For treatment three, Penmarked, the markers use the digital copy of the assignment with Penmarked and the IDE.

Different information was returned to students for each treatment. For paper they received the annotated paper copy of the assignment, and a paper, freehand completed marking schedule. For the database treatment they were emailed a simple list of the marking schedule with their mark for each item and any text comments alongside the item that the marker had entered into the database. For the Penmarked treatment the student received an email with an annotated pdf attached. The first page of the pdf showed the completed marking schedule followed by the digital ink annotated assignment. In all cases the completed marking schedule was returned and the students' total mark was available on the student management system.

We collected the following data for each assignment: assessment number, treatment, marker, marking time, number and types of annotations, grades, and the number of marking appeals/complaints. After the three assignments had been marked we surveyed marker's opinions, and student's opinions.

The marking times were recorded by each marker as they marked. TAs are paid by the hour for marking based on an estimate of the time required for marking. We agreed with the TAs on a fix, generous, number of hours before the study commenced to negate time pressures on them for the task. An adapted annotation categorization system from (Marshall 1997) was used to categorize annotations as either tick or cross, comment, grade or other. Grades were taken from marked assignments. The TAs' and teachers' opinions were gathered through semi-structure interviews and student opinions were garnered from an on-line survey. Student complaints and appeals were recorded and matched back to the assessment number, treatment and marker.

6 Results

Table 1 summarizes the results of the quantitative data on marking times, grades and annotations. We ran a series of statistical tests against this data to identify significant variations. First we compared each set of grades using a one-way ANOVA, these showed no significant differences between markers or groups for each assessment (p .298, p.327, p .265).

Analysis of the marking times showed significant differences for all three assessments. Comparing individual markers there was one marker who was significantly slower through all three treatments. However, disregarding this summary statistics showed that paper marking was significantly slower than the other treatments for the first assessment (<p .001). With assessment 2 there was no significant time difference between the paper and database marking but a significant difference between these two and Penmarked (p .002). With the third assessment Tukey HSD test showed there are significant differences between each treatment (p values between .03 and .004). Individual differences between markers account for some of this difference, however it is clear that Penmarked was consistently faster for markers than either of the other treatments.

Similar tests were conducted on the total number of annotations and number of annotated assignments. The differences were significant in all cases ($p < .01$) except

the first assessment where there was no significant difference between the paper and Penmarked treatments.

Notably for the first assessment while 100% of the assignments marked with the paper and Penmarked treatments had annotations less than 20% of the database treatment had any individual comments and indeed our analysis of these showed most of the comments added to the database were simply 'not implemented' for one marker, or 'not there' for the other. Similar simple comments were evident in the database for the second and third assessments. Our suspicion is that these were pasted in by the markers.

We had not anticipated looking at where the annotations were on the documents however, with assessment 3 when analysing the paper copies it became obvious that all the annotations were on the front or back page – absolutely none were placed in-situ. The Penmarked annotations displayed some similarities with some annotations placed at the top of a class rather that close to the procedure they were related to. Comparing the total number of annotations and assignments with annotations across the assignment we note a general decline in the annotations between the assessments. A number of reasons could contribute to this, for example the students are less likely to make syntactic or layout mistakes. Or it could be as the programs get bigger it is more difficult to identify the critical parts of the code for the ink annotation, particularly with in the paper treatment.

			Time		Grades		Annotations	
Assessment	Treatment	number of assignment	mean	std	mean	std	total	assignments with annotations
1	Paper	60	20.9	10.6	78.3	16.2	275	60
	Database	58	10.3	3.2	82.2	16.6	77	11
	Penmarked	58	9.3	4.4	82.3	15.6	264	58
2	Paper	56	22.3	8.3	71.9	20.9	344	28
	Database	51	21.0	6.9	61.9	23.9	83	16
	Penmarked	56	16.1	4.3	64.4	23.2	490	24
3	Paper	53	21.4	3.8	79.5	20.1	33	5
	Database	58	26.7	12.6	78.1	23.3	91	17
	Penmarked	52	18.6	4.7	82.9	19.6	269	23

Table 1 Summary of quantitative marking data

We asked the TAs to rank the treatments on preference, speed and accuracy. They all ranked Penmarked first, database second and paper third. Their comments supported this very strongly. They appreciated the work-flow support Penmarked afforded telling us that the start-up and close down time for each assignment was considerably less with Penmarked. They particularly commented on the one tap access to the source files from the folder icon. They also compared annotating in Penmarked favourably with annotating the paper; the eraser in Penmarked was the winner here! As they tended to mark at home or when they had gaps between their own class commitments they found carrying around a bundle of paper a nuisance. Four of the markers had used the database treatment when marking for the same course the previous semester and two of these had used another similar tool. Again they all expressed a preference for Penmarked, commenting that it was more natural and easier to comment directly on the assignments. Two express concerns about their spelling and would have like the ability to spell-check their handwritten comments.

The teachers also made complementary comments about Penmarked. While we had recorded grading appeals, they often answered ad hoc questions about marking. They found they were getting less questions from the annotated assignments. Another benefit of Penmarked, that we had not considered is that the teacher had a complete digital replica of what was returned to the student. They found this useful for producing copies of marked work for course review or external moderation. One also commented that she had, on a couple of occasions had students change a handwritten grade on their work and then claim it had been added/recorded incorrectly. Having a digital replica would stop this type of dishonestly.

Question	Treatment	Answers		
I found the submission easy		Agree	Neutral	Disagree
	Paper	6	14	17
	Database	30	8	1
	Penmarked	23	13	1
I found the return of work easy		Agree	Neutral	Disagree
	Paper	12	9	16
	Database	28	11	0
	Penmarked	16	12	9
I understood where I had lost or gained marks		Mostly	Some-times	Rarely
	Paper	19	7	11
	Database	13	19	7
	Penmarked	18	15	4
The feedback helped my learning		A lot	A bit	Not at all
	Paper	7	16	14
	Database	2	27	9
	Penmarked	10	21	6
From the feedback I know how to correct my mistakes		Completely	A bit	Not at all
	Paper	8	15	14
	Database	8	22	8
	Penmarked	7	21	9
I prefer to get my assignments marked this way		Completely	A bit	Not at all
	Database	16	17	5
	Paper	6	12	19
	Penmarked	17	13	9

Table 2 Student Survey

We also ran a voluntary on-line surveyed for the students. Only 39 of the 200 students responded, not enough from which to draw any firm conclusions. A summary of the questions and responses is shown in Table 2. The first part of the questionnaire was to find out what they actually did with the assignments. The first two questions on the convenience of the different methods we added because of their grumblings about the paper treatment. If this sample is representative of the class it is clear from their responses that they disliked having to hand-in and collect paper with over 30% of them not collecting the marked assignment. This was consistent with a large pile of uncollected paper copies at the end of the course. They received the other marking feedback as an email attachment. Notably almost all claim to have read the comments on the paper or pdf, but only about half read the database comments.

As the purpose of feedback is to aid learning, if they had not looked at the markers comments the feedback would not have contributed to their learning. The next set of questions was to elicit their opinion on the contribution to learning. Because of the way the data was collected we could not exclude answers from the respondents that had not collected their paper from this set of answers. They said they understood where they had lost and gained marks better with feedback on the Penmarked treatment and considered that it had helped their learning more. However there was no difference between any treatments on their ability to correct mistakes.

Assessment Treatment	1	2	3
Paper	2	3	3
Database	3	5	7
Penmarked	1	0	2

Table 3 Number of Complaints and Appeals

The final data we collected was the number of complaints and appeals for remarking. The numbers are small so no firm conclusions can be drawn from them. However it does reinforce the data from the student survey with them being more satisfied or better understanding the marks allocated when the assignment has been annotated.

The hypothesis that marking time would not increase with Penmarked was proved, in fact Penmarked was

shown to be the fastest marking treatment. Grades remained consistent across all treatments; we did not take any extraordinary measure to ensure this consistency so feel confident that this is a valid result. We found that Penmarked did encourage markers to comment more on the students work in all cases except when comparing Penmarked to the paper copies of the first assignment. The difference in this case is that the paper copies are only 1-2 pages. The data from the markers, students and teachers is qualitative and suggested increased satisfaction from all parties. This is supported by the decrease in complaints and appeals for the annotated assignments.

7 Discussion

Our initial goal with this project was to create a software environment for marking programs that would support red ink annotation in a paper like manner. At the same time we did not want to adversely affect the work processes around marking. From a practical perspective we did not want to increase the time required for marking assignments. Nor could the software skew the grades in any way. The primary goal was to increase feedback and learning for the students, the aims were to increase their learning by increasing understanding of how they had gained or lost marks and how they could perform better, while at the same time increase their satisfaction with the grades they received.

Penmarked has been developed over three iterations. The experience we have gained from repeated use of the system ourselves and by other teachers and through the more formal usability studies resulted in interesting changes to the system. The area that has undergone the most change is the student list. Initially it was a simple list, we then added check boxes and later a right-click menu and colour coding for incomplete assignments or those to be revisited. We conclude from this that the work-flow support that this list provides is essential to the success of Penmarked.

In contrast the only alteration made to the annotation pane is the addition of a 'find' function. Users have suggested syntax highlighting and a clipboard for comments. The suggestion for 'find' and syntax highlighting lead to our investigations into implementing similar functionality into an IDEs {Chang, 2008 #514}. To date we have had limited success with IDEs: Visual Studio 2010 may make this easier and is worthy of investigation. The clipboard for comments is, at first glance, an obvious extension. However there are significant technical and user challenges to pasting ink, it is not as easy to reflow into a space as text. Our suspicion is that it would take longer to find and place the clipboard comment than to rewrite it. We also worry that a clipboard would encourage lazy generic comments evident in the database treatment in the study above. As educationalists we see little value in this type of comment.

We undertook some informal usability tests on the marking schedule when it was first developed to set screen sizes and time delays. It has remained unchanged since then. Clearly there are many other types of marking scales used; A, B, C …; Likert scales, Excellent – Unsatisfactory; and so on. Supporting other scales may need some redesign of the interface but we would not anticipate any great difficulties.

The responses from students and reduced complaints suggest that annotating their work is worthwhile. We would have like to have seen their confidence to correct mistakes increase. This is an area that could do with further investigation.

The negative comments from the TAs and students about paper suggest to us that for this generation paper and program code are incompatible. Both parties were more comfortable with the electronic systems. It should be noted that this is a programming course; hence we would expect the students and TAs to be at ease with computers. Using this system with a liberal arts class may elicit a different response.

Reflecting on the project ourselves the most important lessons we have learnt have been about the requirements for supporting the entire activity. In our institute paper copies of programming assignments have not been used for many years. Yet we needed to go back to understand the role of paper documents in marking assignments (Marshall 2003) in order to provide the equivalent functionality in a computer system. At the same time it was essential that we provided an easy interface to the existing institutional systems that support student management.

Most of the functionality of Penmarked is available in other software tools, annotation is available in commercial packages such as Microsoft Word or Adobe Acrobat, marking databases are easy to construct and exist in many forms. The display of program code is better in IDEs than that provided in Penmarked. The essential benefit of Penmarked is the bringing together of these different functions in a manner specifically designed to support program marking.

8 Conclusions

Penmarked is a specific example of a wider problem: How to translate successful paper-based techniques to a paperless system without compromising best practice. We have demonstrated that Penmarked is more time efficient and produces better results than either a paper-based system or a marking database. At the same time student understanding and satisfaction increased. The success of this project, we believe, is due in main to our rigorous efforts to support the whole process of marking an assessment from setting up the marking schedule and collecting the assignments to the return of work and filing of grades.

There are areas of Penmarked that could be improved; in particular we would like to include syntax highlighting. An alternative approach is to implement the same functionality into a programming IDE. In an IDE annotation could also be used for code review and for programmers to keep notes for themselves and others. Developing the annotation functionality inside and IDE has proved to be technically difficult. We hope that as pen-based computing becomes more common place that more basic controls support annotation and transparent overlays.

Many organizations are attempting to 'go paperless'. Our experiences with this project suggest that they will have a higher success rate if careful consideration is

given to all tasks related to the activity to be supported.

9 References

Chang, S. H.-H., X. Chen, et al. (2008). Issues of Extending the User Interface of Integrated Development Environments. Chinz, Wellington, ACM.

Chen, X. and B. Plimmer (2007). CodeAnnotator: Digital Ink Annotation within Eclipse. OzCHI 2007: Entertaining User Interfaces Adelaide, 211- 214, ACM.

Francik, E. (1995). "Rapid, integrated design of a multimedia communication system." Human Computer Interface Design 36-69.

Goldschmidt, G. (1999). The backtalk of self-generated sketches. Visual and spatial reasoning in design. J. S. Gero and B. Tversky. Sydney, Key Centre, University of Sydney: 163-184.

Golovchinsky, G. and L. Denoue (2002). Moving markup: repositioning freeform annotations. UIST '02, Paris, France, 21-30, ACM.

Greenaway, R. (2003, 24/6/2006). "Experiential Learning Cycles." Retrieved 24/6/03, 2003, from http://reviewing.co.uk/research/learning.cycles.htm.

Heinrich, E. and A. Lawn (2004). Onscreen marking support for formative assessment. Ed-Media, 1985-1992.

Heinrich, E., Wang, et al. (2003). Online Marking of Essay-type Assignments. Ed-Media, Norfolk, USA, 768 - 772.

Marshall, C. (1997). Annotation: from paper books to the digital library. DL, Philadelphia, 131-140, ACM.

Marshall, C. (2003). Reading and Interactivity in the Digital Library: Creating an experience that transcends paper. CLIR/Kanazawa Institute of Technology Roundtable, Kanazawa, Japan, 5.4.1-20.

Marshall, C. C., M. N. Price, et al. (2001). Designing e-Books for legal research. JCDL '01, Roanoke, Virginia, 41-48, ACM.

Mason, P. and B. Plimmer (2005). A Critical Comparison of Usability Testing Methodologies. NACCQ, Tauranga, 255-258, NACCQ.

Myers, D., E. Hargreaves, et al. (2004). Developing Marking Support within Eclipse. OOPSLA, 62 - 66

Plimmer, B. and P. Mason (2004). Designing an Environment for Annotating and Grading Student Assignments. OZCHI, Wollongong, 45-53.

Plimmer, B. and P. Mason (2006). A Pen-based Paperless Environment for Annotating and Marking Student Assignments. AUIC, Hobart, 27-34, CRPIT.

Priest, R. and B. Plimmer (2006). RCA: Experiences with an IDE Annotation Tool. CHINZ, Christchurch, 53-61, ACM.

Ramachandran, S. and R. Kashi (2003). An architecture for ink annotations on web documents. 17th International Conference on Document Analysis and Recognition, 256-260, IEEE Computer Society.

Schilit, B. N., G. Golovchinsky, et al. (1998). Beyond Paper: Supporting active reading with free form digital ink annotations. CHI 98, Los Angeles, CA, 249-256, ACM.

Sellen, A. J. and R. H. R. Harper (2002). The myth of the paperless office. Cambridge MA, MIT.

Shipman, F., M. Price, et al. (2003). Identifying useful passages in documents based on annotation patterns. ECDL, Trondheim, Norway, 101-112.